Oracle® Configurator Developer

User's Guide

Release 11*i*

Part No. B13603-03

May 2005

This document describes how to build and deploy configuration models using Oracle Configurator Developer.

Oracle Configurator Developer User’s Guide, Release 11*i*

Part No. B13603-03

Primary Author: Stephen R. Damiani

Contributing Author: Tina Brand

Contributor: Mark Sawtelle, Harriet Shanzer, Oracle Configurator Development Group

Contents

List of Figures

List of Tables

Send Us Your Comments

Oracle Configurator Developer User's Guide, Release 11*i*

Part No. B13603-03

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: czdoc_us@oracle.com
- FAX: 781-238-9896 Attn: Oracle Configurator Documentation
- Postal service:

  Oracle Corporation
  Oracle Configurator Documentation
  10 Van de Graaff Drive
  Burlington MA 01803-5146
  USA

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Welcome to the *Oracle Configurator Developer User's Guide*. This user's guide includes the information you need to work with Oracle Configurator Developer effectively.

Intended Audience

This guide assumes you have a working knowledge of your business processes, tools, and configurations. It also assumes you are familiar with configurator applications. If you have never used a configurator application, we suggest you attend one or more of the Oracle Configurator training classes available through Oracle University. You should also be familiar with Oracle Applications and the Oracle Applications database.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Structure

This user's guide contains a table of contents, lists of examples, tables and figures, a reader comment form, several chapters, two appendixes, a glossary, and an index. The chapters are organized in six parts. Within the chapters, information is organized in numbered sections of several levels. Note that level does not imply importance or degree of detail. For instance, third-level sections in one chapter (x.x.x) may not contain information of equivalent detail to the third-level sections in another chapter.

- Part I, "About Configuration Models"
  - Chapter 1, "Introduction" includes information on planning your project to ensure success using Oracle Configurator Developer.
  - Chapter 2, "The CZ Schema's Item Master" describes the schema within the Oracle Applications database that stores Oracle Configurator Developer's data.
  - Chapter 3, "Types of Models" defines the term "configuration model" and presents the types of Models that are available in Oracle Configurator Developer.
  - Chapter 4, "References" describes Model References, including how they are used in BOM Models, configuration rules, and User Interfaces.
  - Chapter 5, "Properties" describes Properties, their relationship to Model structure, and how they can be used when defining rules.
  - Chapter 6, "Effectivity" describes how to use effectivity when building configuration models and defining rules.
  - Chapter 7, "Instantiation" describes how to build a configuration model that allows Oracle Configurator end users to create multiple instances of configurable components at runtime.
  - Chapter 8, "Connectivity" includes information about creating configuration models in Configurator Developer that allow components to be connected in a runtime Oracle Configurator.
- Part II, "Model Structure"
  - Chapter 9, "Model Structure Node Types" describes the various kinds of Model structure nodes available when building a configuration model in Configurator Developer.
  - Chapter 10, "Using Populators"describes how to use Populators to create Model structure using data in the CZ schema's Item Master.
- Part III, "Configuration Rules"
  - Chapter 11, "Rule Basics" provides information on defining rules that determine what options an end user can select to create a valid configuration.
  - Chapter 12, "Logic Rules" describes logical relationships and the different types of Logic Rules you can create in Configurator Developer.
  - Chapter 13, "Numeric Rules" presents general information about Numeric Rules, such as how they work and ways you can use them when building a configuration model.
  - Chapter 15, "Comparison and Compatibility Rules"describes how you can use these types of rules when building a configuration model.

  - Chapter 14, "Design Charts" describes how you can use this type of rule when building a configuration model.
  - Chapter 16, "Statement Rules" describes how to create a rule in Oracle Configurator Developer that is written in the Constraint Definition Language (CDL).
  - Chapter 17, "Configurator Extensions" describes how to create a rule in Oracle Configurator Developer that extends a runtime Oracle Configurator using Java code.
  - Chapter 18, "Rule Sequences" describes Rule Sequences and how you can use them when building a configuration model.
- Part IV, "Runtime User Interfaces"
  - Chapter 19, "Displaying the Model" presents general information about how a User Interface that you generate in Configurator Developer displays Model structure nodes and other data.
  - Chapter 20, "User Interface Templates" describes the templates you can use to generate a User Interface in Configurator Developer.
  - Chapter 21, "User Interface Structure and Design" provides information about the structure of a generated User Interface and the various types of UI elements that are available.
- Part V, "Testing and Publishing"
  - Chapter 22, "Testing and Debugging" describes how to unit test a configuration model.
  - Chapter 23, "Publishing" provides information about publishing, which is typically the final phase of the configuration model development process.
- Part VI, "Developer Tool Reference"
  - Chapter 24, "Configurator Developer User Interface Basics" provides general information about the Configurator Developer user interface.
  - Chapter 25, "Main Area of the Repository" presents general information about the objects you can create and maintain in this area of the application, such as Models, Folders, Effectivity Sets, and Usages.
  - Chapter 26, "Item Master Area of the Repository" describes the various tasks you can perform in this area of the application, such as creating and modifying Items and Item Types.
  - Chapter 27, "Publications Area of the Repository" describes the various tasks you can perform in this area of the application, which include creating and maintaining Model publications.
  - Chapter 28, "General Area of the Workbench" describes the information provided and tasks you can perform in this area of the application. For example, generating Model logic, refreshing User Interfaces, and modifying a Configurator Extension archive path.
  - Chapter 29, "Structure Area of the Workbench" describes tasks you can perform in this area of the application, such as creating Components, Features, Options, References, and so on.
  - Chapter 30, "Rules Area of the Workbench" explains how to create and maintain each type of configuration rule.

  - Chapter 31, "User Interface Area of the Workbench" explains how to create and modify User Interfaces that you generate in Oracle Configurator Developer.
  - Chapter 32, "Model Debugger and User Interface Testing" describes how to unit test a configuration model using either the Model Debugger or a generated User Interface.
- Part VII, "Appendices"
  - Appendix A, "The Runtime Oracle Configurator" describes tasks commonly performed when configuring an item in a runtime Oracle Configurator.
  - Appendix B, "Multiple Language Support" describes Multiple Language Support (MLS) and things to consider when implementing MLS with Configurator Developer and a runtime Oracle Configurator.
  - Appendix C, "Rules, Node Types, and System Properties" lists which Model structure nodes and System Properties are valid when defining a Logic, Numeric, or Comparison Rule.
- The "Glossary" contains definitions that you may need while working with Oracle Configurator.

Related Documents

For more information, see the following manuals in Release 11*i* of the Oracle Configurator documentation set:

- *Oracle Configurator Installation Guide*
- *Oracle Configurator Implementation Guide*
- *Oracle Configurator Performance Guide*
- *Oracle Configurator Modeling Guide*
- *Oracle Configurator Extensions and Interface Object Developer's Guide*
- *Oracle Configurator Constraint Definition Language Guide*

Additionally, refer to the following manuals in Release 11*i* of the Oracle Applications documentation set:

- *Oracle Bills of Material User's Guide*
- *Oracle Inventory User's Guide*
- The Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1)

Conventions

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The table below lists other conventions that are also used in this manual.

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.

Convention	Meaning
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted
boldface text	Boldface type in text indicates a new term, a term defined in the glossary, specific keys, and labels of user interface objects. Boldface type also indicates a menu, command, or option, especially within procedures
italics	Italic type in text, tables, or code examples indicates user-supplied text. Replace these placeholders with a specific value or string.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
>	The left bracket alone represents the MS DOS prompt.
$	The dollar sign represents the DIGITAL Command Language prompt in Windows and the Bourne shell prompt in Digital UNIX.
%	The percent sign alone represents the UNIX prompt.
`name()`	In text other than code examples, the names of programming language methods and functions are shown with trailing parentheses. The parentheses are always shown as empty. For the actual argument or parameter list, see the reference documentation. This convention is *not* used in code examples.
&	Indicates a character string (identifier) that can display text dynamically in Configurator Developer or a runtime Oracle Configurator. For example, "&PROPERTY" can be used to dynamically construct and display a Property of a Model node.
|_	Used in graphics that show Model structure to indicate a parent-to-child relationship between two nodes.
|->	Used in graphics that show Model structure to indicate a Model Reference node.
|~>	Used in graphics that show Model structure to indicate a Connector node.

Getting Help with Oracle Configurator Developer

Online help is available from Oracle Configurator Developer via the global help link and the content is the same as the *Oracle Configurator Developer User's Guide* and the *Oracle Configurator Constraint Definition Language Guide*.

Product Support

The mission of the Oracle Support Services organization is to help you resolve any issues or questions that you have regarding Oracle Configurator Developer and Oracle Configurator.

To report issues that are not mission-critical, submit a Technical Assistance Request (TAR) using Metalink, Oracle's technical support Web site, at:

```
http://www.oracle.com/support/metalink/
```

Log into your Metalink account and navigate to the Configurator TAR template:

1. Choose the **TARs** link in the left menu.
2. Click on **Create a TAR**.

3. Fill in or choose a profile.
4. In the same form:
   a. Choose **Product**: Oracle Configurator or Oracle Configurator Developer
   b. Choose **Type of Problem**: Oracle Configurator Generic Issue template
5. Provide the information requested in the iTAR template.

You can also find product-specific documentation and other useful information using Metalink.

For a complete listing of available Oracle Support Services and phone numbers, see:

`http://www.oracle.com/support/metalink`

Troubleshooting

Oracle Configurator Developer and Oracle Configurator use the standard Oracle Applications methods of logging to analyze and debug both development and runtime issues. These methods include setting various profile options and Java system properties to enable logging and specify the desired level of detail you want to record.

For information about logging while working in Configurator Developer, see Section 24.3.5, "Diagnostics" on page 24-11.

For details about the logging methods available in Configurator Developer and a runtime Oracle Configurator, see:

- The *Oracle Applications System Administrator's Guide* for descriptions of the Oracle Applications Manager UI screens that allow System Administrators to set up logging profiles, review Java system properties, search for log messages, and so on.
- The *Oracle Applications Supportability Guide*, which includes logging guidelines for both System Administrators and developers, and related topics.
- The Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

Part I

About Configuration Models

Part I describes the different types of configuration models you can create and presents some general Model development techniques and concepts.

Part I contains the following chapters:

1 Introduction

This chapter provides general information about Oracle Configurator Developer and includes the following sections:

- Oracle Configurator Developer
- The Runtime Oracle Configurator
- The Overall Process

1.1 Oracle Configurator Developer

Oracle Configurator Developer is an Oracle Applications product that enables you to rapidly develop a configuration model and a configurator. Configuration model is defined in Section 3.1 on page 3-1.

A **configurator** is the part of an application that provides custom configuration capabilities. A configurator is usually launched from a host application, such as Oracle Order Management or *i*Store, and displays the selected configuration model to the end user. During an Oracle Configurator session, an end user makes selections and specifies requirements for the product or service being configured. Oracle Configurator collects the customer's requirements and, using the Model definition and rules you defined in Configurator Developer, ensures that the end user creates a valid **configuration**.

A configurator can be thought of as a selling tool. The Model bill serves as a guide to selecting configuration options. A configuration created during an Oracle Configurator session is based on an already existing Model bill and results in a standard manufacturing bill of materials. Configurations do not have to be based on existing Model bills of material, although that is currently necessary for ordering and downstream ERP applications.

1.1.1 Launching Oracle Configurator Developer

You begin an Oracle Configurator Developer session by logging into Oracle Applications and then choosing a responsibility that provides access to Configurator Developer. (The predefined Oracle Configurator Developer responsibilities are described in the *Oracle Configurator Implementation Guide*.) You then select **Oracle Configurator Developer** from the list of available applications.

Oracle Configurator Developer consists of a **Repository** and a **Workbench**. These areas provide the tools you use when creating and maintaining configuration models.

1.1.2 Repository

Use the areas of the Configurator Developer Repository to organize Models and manage objects such as Effectivity Sets, Usages, UI Templates, Items and Item Types, Properties, Configurator Extensions, and Model Publications.

For more information about each area of the Repository, see Part VI, "Developer Tool Reference".

1.1.3 Workbench

The different areas of the Configurator Developer Workbench provide tools for creating, modifying, and testing Model structure, configuration rules, and UI definitions. In the User Interface area, for example, you can generate a User Interface that is based on the Model structure, and then edit it to meet your product's unique requirements.

For more information about each area of the Workbench, see Part VI, "Developer Tool Reference".

1.1.4 Hierarchical Structure

Oracle Configurator Developer displays many objects, such as the Model, configuration rules, and a generated User Interface, in a hierarchy. This structure shows how elements are related to each other and indicates which objects contain other objects. When an object contains other objects, a **parent and child** relationship exists between them. For example, within a Model, Component A contains Feature X, Y, and Z. In this relationship, Component A is the parent and the Features are its children.

In this user's guide, each object within the Model structure is called a node. The node at the top of this structure is always a Model, and is called the root node. The Rules area of the Workbench and the User Interface area of the Workbench also display objects in a hierarchy (for example, rules, Folders, and UI elements), to indicate how they are organized and their relationship to other objects.

By default, Configurator Developer displays hierarchical data in a "collapsed" state, so only the root of the structure and the first level of nodes beneath the root are visible. You can expand or collapse sections of the hierarchy using the plus (+) and minus (-) controls, or apply the action to the entire structure by clicking Expand All or Collapse All. In the Main area of the Repository, for example, you can expand any Folder that contains one or more Models, Effectivity Sets, Usages, or other Folders. Within the each area of the Workbench, these controls appear next to any node that has children. For example, a Component that contains Features, or BOM Option Class that contains BOM Standard Items.

1.2 The Runtime Oracle Configurator

The fundamental elements of a configurator built with Oracle Configurator Developer are:

- Model structure that organizes the parts of your product, such as an imported BOM Model
- Configuration rules that constrain the relationships among parts of your product
- A User Interface (UI) that optionally reflects the Model structure, enables end users to interact with the configuration model, and defines the appearance of the runtime Oracle Configurator

Model structure, rules, and it UI(s) are stored in the **CZ schema**, which is a sub-schema of the Oracle Applications database (for details, see Chapter 2). The compiled configuration rules and Model structure exist as the generated logic in the CZ schema. This logic enforces valid configurations based on end-user selections. The User Interface definitions of the configuration model function as the runtime Oracle Configurator **User Interface**. The User Interface (UI) also interprets the data in the CZ schema and keeps the UI state current as the end user makes selections. In other words, when the end user configures an item in the runtime Oracle Configurator, the CZ schema, configuration rules, and the User Interface determine what is available for selection, what results from selections, and how the configuration model is displayed.

Oracle Configurator is integrated with Oracle Applications so that an end user can configure a product based on a Bill of Materials (BOM). Oracle Configurator dynamically creates a configuration model that reflects BOM Model rules, including parent-child, optional or required selections, mutually exclusive selections, and Quantity Cascade rules. In this case, the BOM Model is neither imported into the CZ schema nor published from Configurator Developer; an end user configures the BOM Model using the Generic Configurator User Interface. For more information about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

If you want to add additional Model structure to a BOM Model, define additional rules, and generate a User Interface, you must do so in Configurator Developer. In this case, you can deploy a runtime Oracle Configurator by generating and customizing an HTML-based User Interface in Configurator Developer. For more information, see Chapter 31.

Refer to the *Oracle Configurator Implementation Guide* for more information on the mechanics of deploying a runtime Oracle Configurator.

1.3 The Overall Process

Implementing and maintaining an Oracle Configurator consists of the following steps:

1. Complete Oracle Configurator Developer training.

   Ask your Oracle representative about training classes available through Oracle University.

2. Read the *Oracle Configurator Modeling Guide* to become familiar with recommended best practices when designing Model structure and configuration rules.
3. Plan your project. See Section 1.3.1 on page 1-4.
4. Set up Configurator Developer. See Section 1.3.3 on page 1-4.
5. Build a configuration model in Configurator Developer. This step typically includes extending an imported BOM Model by adding structure, defining configuration rules, and creating one or more User Interfaces. See Section 1.3.4 on page 1-5.
6. Unit test the configuration model in Configurator Developer. See Section 1.3.4.3 on page 1-6.
7. Deploy the configuration model. This step includes:
   - Publishing the configuration model
   - System testing the configuration model

   See Section 1.3.5 on page 1-6.
8. Manage configuration models and publications. See Section 1.3.6 on page 1-7.

1.3.1 Plan your Project

Plan carefully before beginning to build a configuration model in Oracle Configurator Developer. Consider the following points during planning:

- Design your configuration model and consider what functionality your end users require. For example, you may need to add a customer needs assessment component to your Model. For more information, see the *Oracle Configurator Modeling Guide*.
- Plan to express your requirements for valid configurations in terms of the rules that Configurator Developer provides. See Section 1.3.4.1, "Design Configuration Rules" on page 1-5.
- Establish standardized and meaningful naming conventions for Model nodes and rules.
- Plan your User Interface. Configurator Developer provides several UI Master Templates and UI Content Templates that you can use to generate a User Interface. You can also perform many customizations to a generated UI using the User Interface area of the Workbench. For details, see Part IV, "Runtime User Interfaces".

  If you need to deploy configuration models in multiple languages, consider the requirements for implementing Multiple Language Support (MLS). See Appendix B.
- For custom deployments that are not integrated with Oracle Applications:
  - Gather the requirements for needed outputs such as quotes, proposals, and order entry data, their format, and the data for populating them. Oracle Configurator provides pre-defined output for Oracle ERP orders.
  - Gather the requirements for integrating your system with other systems such as data synchronization and replication, quotes, and orders.
- Develop a plan for publishing configuration models. For details, see the *Oracle Configurator Implementation Guide*.

1.3.2 Identify your Product Data

To build configuration models using Configurator Developer, you may want to use enterprise data from Oracle Inventory and Oracle Bills of Material, or a legacy system. You populate the CZ schema's **Item Master** in the Oracle Applications database with data from Oracle Bills of Material by running a concurrent program. For more information about the CZ schema's Item Master, see Section 2.1 on page 2-1. The import process is explained in the *Oracle Configurator Implementation Guide*.

If your data comes from Oracle Inventory Items and Oracle Bills of Material, or from an external data source, you must develop a mechanism for populating the Configurator import tables, and a plan for refreshing the import as required. Your Database Administrator (DBA) may prepare existing enterprise data for import. To import data from a legacy system, see the *Oracle Configurator Implementation Guide*.

You can also populate the CZ schema's Item Master from Configurator Developer by manually creating Items, Item Types, and Properties. For details, see Section 29.6 on page 29-4.

1.3.3 Set Up Oracle Configurator Developer

Oracle Configurator Developer is an Oracle Applications product and is installed along with other applications in the Oracle E-Business Suite by running Rapid Install.

Rapid Install also provides default values for all profile options and Oracle Configurator servlet properties. For details, see the *Oracle Configurator Installation Guide*.

The user name you enter when logging into Oracle Applications must be assigned to at least one of the responsibilities that provides access to Oracle Configurator Developer. Defining Oracle Applications users is described in the *Oracle Applications System Administrator's Guide*. For a list of the predefined Configurator Developer responsibilities, see the *Oracle Configurator Implementation Guide*.

To ensure that the CZ schema contains the data you need, see the *Oracle Configurator Implementation Guide*, or your DBA.

If you need to maintain User Interfaces created in a previous version of Configurator Developer, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.

Note: If you maintain both a development and a production database, do not run Configurator Developer in your production instance. For more information, see the *Oracle Configurator Implementation Guide*.

1.3.4 Build a Configuration Model

There are two approaches to creating a configuration model in Oracle Configurator Developer:

- Self-contained mode
- Integrated mode

In both modes, you build a configuration model based on item structure and data. If you are working in the self-contained mode, you create the Item Master and build your Model, configuration rules, and User Interface entirely within Configurator Developer. You might choose to work this way if you are building a small-scale demonstration or prototype system.

Many real-world configuration models involve working in the integrated mode. Configuration models that are created in integrated mode are based on products defined in Oracle Bills of Material. You import the BOM Model into the CZ schema, optionally build Model structure and define rules in Configurator Developer, and then deploy the configuration model. After an Oracle Configurator end user configures the item, it is passed on to Oracle Order Management for order fulfillment and downstream processing by other Oracle Applications products. For details about the data import process, see the *Oracle Configurator Implementation Guide*.

1.3.4.1 Design Configuration Rules

Carefully consider what rules you need to build into your configuration model. The design step may include writing a functional specification and other design documents.

When you define the requirements for your configurator, you define the rules that make default selections, constrain options based on other selections, and guide end users in creating a valid configuration. You now need to determine how you can most effectively and efficiently apply these rules, using the kinds of configuration rules that Oracle Configurator Developer provides.

Ask yourself questions such as:

- What components must be included in a valid configuration?
- What components are optional?
- What components are compatible with each other?
- What selections affect another selection?
- What are valid initial selections?
- What rules define the configuration of product families?
- What rules define the relations among product families?

Refer to the following documentation for additional things to consider when defining rules:

- *Oracle Configurator Modeling Guide*
- *Oracle Configurator Performance Guide*

For information about defining rules, see Chapter 11.

1.3.4.2 Create a User Interface

After building Model structure and rules, generate a User Interface to view and unit test the Model in a runtime Oracle Configurator. If necessary, you can generate a variety of User Interfaces from a single Model's structure, or create a User Interface that is not based on the Model's structure. Generating a User Interface is described in Section 31.2 on page 31-1.

1.3.4.3 Unit Test the Configuration Model

Click the **Test Model** button to unit test a configuration model periodically during development. This button appears in all pages of the Structure and Rules areas of the Workbench and enables you to either unit test a User Interface created in Configurator Developer, or run the **Model Debugger**.

For more information about unit testing, see Chapter 22, "Testing and Debugging".

After unit testing and model development is complete, deploy the configuration model for integration and system testing. See Section 1.3.5, "Deploy the Configuration Model" on page 1-6.

1.3.5 Deploy the Configuration Model

Deploying a configuration model requires integration with other applications and rigorous system testing before making it available to customers in your production environment.

1.3.5.1 Integration

Oracle Configurator can be called from many different host applications. For a complete list of applications that support Oracle Configurator, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle’s technical support Web site.

If the Oracle Configurator is embedded in an Oracle Applications product (such as Order Management or *i*Store), there might not be any additional setup required after running the Oracle Applications Rapid Install process. However, some additional setup may be required to launch the embedded configurator in a non-Oracle Applications product. For more information, see the *Oracle Configurator Installation Guide*.

If the Oracle Configurator is embedded in a custom Web application, the host application must generate the initialization and termination messages that start and stop the embedded configuration session. See the *Oracle Configurator Implementation Guide* for more information.

1.3.5.2 Testing

A configuration model itself should be periodically unit tested while building it in Configurator Developer. Unit testing is discussed in Chapter 22. However, because a Model may use effectivity, contain References to other Models, or have multiple UI definitions, a unique permutation of the same configuration model may appear when accessed by a host application. For this reason, you should also thoroughly **system test** your Model for adequate performance, end-user access, security, and any integration customizations before making it available to end users in a production environment. System testing includes **publishing** the configuration model using different applicability parameters and accessing each publication from at least one host application. Publishing is explained in Chapter 23.

1.3.5.3 Production

After unit testing and updating the configuration model in Configurator Developer, and system testing using one or more host applications, make the configuration model available to end users in your production environment.

Before deploying the configuration model in a production environment, test existing configurations against the new version to ensure that users can restore previously saved configurations. If you made changes during the unit testing phase, you can easily bring the new Model on line without interrupting end-user access.

1.3.6 Manage Models and Publications

Managing a Model in Configurator Developer includes updating the Model's structure, rules, and User Interface as your product and business requirements change over time, and updating **Model publications** so your Oracle Configurator end users have access to the most up-to-date configuration model.

You **publish** Models to make a configuration model and UI available to host applications. When a Model is published to your production environment, it becomes the configuration model against which Oracle Configurator end users make selections to configure products and services. You also update existing publications as each configuration model's definition changes over time. For more information, see Chapter 23, "Publishing".

2 The CZ Schema's Item Master

This chapter describes the CZ schema's Item Master.

This chapter includes the following sections:

- Imported Items
- Orderable Items

2.1 Introduction

The runtime Oracle Configurator and Oracle Configurator Developer use the CZ schema within the Oracle Applications database to access and store data. There is only one CZ schema within an Oracle Applications database instance.

The CZ schema contains an **Item Master** subschema. The Item Master is a set of your enterprise data, structured into Items and Item Types, and is the primary source of data for your application. The Item Master consists of **Items**, which are specific elements of a product, and **Item Types**, which are logical groupings of Items. An example of an Item Type is "TV Set", and the available models are the Items within this Item Type, such as 19" Black and White, 21" Color, and 42" Wide Screen. When building configuration models in Oracle Configurator Developer, you can use data in the CZ schema's Item Master to, for example, build Model structure.

In Configurator Developer, Item Master data appears in the Item Master area of the Repository. This area of the Repository is described in Chapter 26.

Note: Do not confuse the CZ schema's Item Master with the Oracle Applications Item Master. In this user's guide, the term Item Master always refers to the CZ schema's Item Master, unless otherwise indicated.

2.2 Imported Items

The Items in the Item Master are either created from scratch in Configurator Developer or are imported from source data in the Oracle Applications Item Master and other tables. In most development situations, Item Master data is imported. The import process is explained in detail in the *Oracle Configurator Implementation Guide*.

Imported data in the CZ schema represents the source data and is only used for defining the configuration model. At runtime, after a configuration has been created and passed back to the host application, items are ordered from the source data.

Legacy data, such as Bills of Material or pricing information, can be imported into the Item Master. Generally, the data source is either an Oracle Applications database or a non-Oracle Applications database. For consistency, imported data should be maintained in the source database. You can see whether an Item was imported by viewing its details page. In other words, click on the item's name in the Item Master area of the Repository, or open it for editing. For information about the changes you can make to Properties when working in the Item Master area of the Repository, see Section 3.3.6.2, "Limitations when Modifying Imported Items, Item Types, and Properties" on page 3-6.

To learn how to create, modify, and delete Items and Item Types, see Chapter 26.

Oracle Configurator Developer provides a mechanism called **Populators** that you can use to create new Model structure from data in the CZ schema's Item Master, or automatically update information to reflect changes to product data. For more information, see Chapter 10, "Using Populators" on page 10-1.

You can view the contents of the Item Master in the Item Master area of the Repository, or by generating a Model Report. For details, see Section 28.1 on page 28-1.

For more information about imported Items and Item Types, see Section 3.3.6, "Item Types and Imported BOM Properties" on page 3-5.

2.3 Orderable Items

When creating an Item or viewing its details in the Item Master area of the Repository, the **Orderable** check box is visible. This setting indicates whether the Item appears in the Configuration Summary page. For details, see Section 19.5, "The Configuration Summary Page" on page 19-11.

The Orderable check box is automatically selected for all imported BOM Items and is read-only in Configurator Developer. By default, this setting is not selected for Items that you create in Configurator Developer, but you can select it if you want the Item to appear in the Configuration Summary page.

Only BOM Items can be ordered from a host application that is part of Oracle Applications (for example, Order Management). Custom implementations may want to use the Orderable setting, for example, to process non-BOM Items downstream in a non-Oracle system.

Note: If the same item appears multiple times in the Model structure (for example, because of a Populator or by creating structure from Items), it will also appear multiple times in the Oracle Configurator Summary page.

3 Types of Models

This chapter defines the term "configuration model" and presents the types of Models that are available in Oracle Configurator Developer.

This chapter includes the following sections:

- What is a Configuration Model
- Models
- Imported BOM Models
- Container Models

To learn about designing a configuration model for an Oracle Configurator, see the *Oracle Configurator Modeling Guide*. For information about optimizing the performance of your runtime Oracle Configurator, see the *Oracle Configurator Performance Guide*.

3.1 What is a Configuration Model

A **configuration model** is Model structure, configuration rules, and optionally a User Interface from which an Oracle Configurator end user makes selections to configure a valid, orderable item. You build configuration models in Configurator Developer based on products or services that can be configured according to validation rules that you define.

Model structure is the hierarchical view of the data that represents the product or service, and is the starting point from which a configuration model is developed. You typically use the Model structure to define configuration rules and generate a runtime User Interface. For details see, Part II, "Model Structure".

3.2 Models

There are two kinds of Models: Models that you create in Configurator Developer, and imported BOM Models. Imported BOM Models are described in Section 3.3 on page 3-2.

Models that you create in Configurator Developer are often created to provide guided buying or selling questions and may reference, or be referenced by, other Models. Guided buying or selling is described in Section 3.2.1 on page 3-2.

Within a Model you can create any type of structure node, including Components, Features, Options and so on. You can also create References to other Models, or to a BOM Model.

Note: In this user's guide, the term "Model" also refers generally to the hierarchical structure of data required to create a configuration model. (In other words, the data that appears in the Structure area of the Workbench.) This structure may consist of a Model that you create in Configurator Developer, an imported BOM Model, or both. To avoid confusion with imported Models, this guide refers to a Model that you create in Configurator Developer as either a non-imported Model or a non-BOM Model.

When viewing a Model in the Structure area of the Workbench, you may notice that a Model that you create in Configurator Developer has a node type of "Component". (You can see this by applying a View that includes the Node Type column. Views are described in Section 24.1.1 on page 24-2.) This is because a Model you create in Configurator Developer has the same characteristics as a Component node; the only difference is that a Model can be referenced by another Model, while Components cannot.

To create a Model, see Section 25.3.1 on page 25-3.

3.2.1 Guided Buying or Selling

Guided buying or selling refers to customer needs-assessment questions that are built into your configuration model to guide and facilitate the configuration process in a runtime UI. It also refers to the Model structure that defines these questions, such as Components, Features, Totals, Resources, and so on, and configuration rules that automatically select some product options and exclude others based on the end user's responses.

For example, in a configuration model for an automobile, you create a Feature whose UI caption is "Select the stereo system you want." The Feature's Options represent the Premium System with 8 speakers and 5CD changer, the Enhanced System with 6 speakers and single CD, and the Basic System with 4 speakers and no CD player. Using these Options, you define rules that select or exclude specific options from the configuration. At runtime, the end user's selections guide the process of configuring a product that best meets their needs.

3.3 Imported BOM Models

Typically, an integrated Oracle Configurator in Oracle Applications is based on an existing BOM Model that is defined in Oracle Bills of Material, and then imported into the CZ schema's Item Master. The integration with Oracle Applications supports the Configure to Order (CTO) process, which includes order entry, demand forecasting, master scheduling, production, shipping, and financial accounting. For details about CTO, see the *Configure to Order Implementation Guide*.

The import process populates the hierarchical Model tree and the CZ schema's Item Master with BOM Model data. Imported BOM data usually includes the root BOM Model and all of its optional components, which may include other BOM Models, BOM Option Classes, BOM Standard Items, and any associated Properties. Imported Properties are discussed in Section 3.3.6 on page 3-5.

Required BOM components are not imported into Oracle Configurator Developer with the BOM Model, since they are not configurable. However, there is one exception to this rule: a required component *is* imported into Configurator Developer if it contains optional components, since in this case the required component is configurable. For more information about optional and required components, see Section 11.3, "Imported BOM Rules" on page 11-3.

Importing BOM Models is described in the *Oracle Configurator Implementation Guide*.

A BOM Model can be ordered from a host application without launching Oracle Configurator to select specific options. In other words, an end user does not have to configure a BOM Model before it can be added it to a sales order and processed by downstream ERP applications. In this case, only required items within the BOM Model are ordered.

Note: Do not confuse BOM components with Oracle Configurator Developer Component nodes. BOM components are Oracle Inventory Items (BOM Models, BOM Option Classes, and BOM Standard Items) that make up the BOM. For more information about Component nodes, see Section 9.6 on page 9-2.

Note: For an important note about the initial logic state of BOM items in the runtime UI, see Section 11.5.3 on page 11-7.

3.3.1 The Imported BOM Model in Configurator Developer

When you import a BOM Model, Configurator creates a corresponding Model node in the top level of the Main area of the Repository (that is, in the root folder). You can then copy or move the Model into a specific folder, or open it for editing in the Workbench.

Each BOM Model imported from Oracle Bills of Material corresponds to one BOM Model in Oracle Configurator Developer. The name of the BOM Model corresponds to the name in Oracle Bills of Material, and the hierarchical structure in Configurator Developer mirrors the BOM Model's structure that is defined in Oracle Bills of Material.

When a BOM Model contains other BOM Models, the child BOM Models appear as Reference nodes in Configurator Developer. For more information, see Section 4.5, "References and BOM Models" on page 4-4.

3.3.2 Types of BOM Models

The types of BOM Models that are configurable include Assemble to Order (ATO) and Pick to Order (PTO) BOM Models. These BOM Models are defined in Oracle Bills of Material using item data defined in Oracle Inventory. The imported data is read-only in Configurator Developer because it must correspond to the BOM Model defined in Oracle Bills of Material throughout the business process. However, you can use Configurator Developer to add Components, Resources, Totals, and so on to meet your configuration requirements.

Note: A PTO BOM Model may also be a **Container Model**. See Section 3.4, "Container Models" on page 3-8.

3.3.3 BOM Model Structure Nodes

Oracle Configurator Developer uses different icons for each type of Model structure node so you can differentiate between imported BOM Model structure nodes and nodes that you create in Configurator Developer. You can also define or modify the View used in the Structure area of the Workbench so it displays the Type column. This column indicates whether each node is a BOM Model, BOM Option Class, non-BOM Model, Component, and so on.

3.3.4 Imported BOM Model Names

In Configurator Developer, the BOM Model name consists of the Model name defined in Oracle Inventory, followed by its Inventory organization ID and Inventory Item ID.

For example:

Production V1 Test (203 52144)

In this example, Production V1 Test is the BOM Model name, 204 is the organization ID, and 52144 is the Oracle Inventory Item ID. (The organization ID and Item ID are *internal* values and are therefore not visible in Oracle Inventory.)

In the Main area of the Repository you can modify the name or description of a BOM Model, and you can change the name of a Reference to a BOM Model in the Structure area of the Workbench, but you cannot modify any information that is imported from Oracle Bills of Material.

3.3.5 Imported BOM Data

When you populate the CZ schema, the following information about each BOM item appears in Configurator Developer:

- **Name**: The name of the item.
- **Description**: A brief description of the item.
- **Definition**: A basic definition of the item, including the BOM Item Type, Minimum and Maximum Quantity, Default Quantity, and whether:
  - The item's optional children are mutually exclusive (see Section 11.3, "Imported BOM Rules" on page 11-3)
  - The item is required in the configuration when its parent is selected
  - The item allows decimal quantities (see Section 3.3.5.1 on page 3-4)
  - The item is Trackable (for details, see the *Oracle Telecommunications Service Ordering Process Guide*)
- **Properties**: Item Catalog Descriptive Elements (Property Names) and Descriptive Element Values (Property Values) that are defined in Oracle Inventory (see Table 3–1 on page 3-5).
- **Property Values**: Item Catalog Descriptive Element values defined in Oracle Inventory.
- **Effective date**: The range of dates in which the item can be added to a configuration.

An imported BOM Model also contains several implicit rules and behaviors that you should understand. For details, see Section 11.3, "Imported BOM Rules" on page 11-3.

3.3.5.1 Decimal Quantities and BOM Items

In Configurator Developer, the Decimal Quantity is Allowed setting appears in the details page for all imported BOM items. (See Section 29.17.5, "Definition" on page 29-12.) This setting indicates whether an Oracle Configurator end user can enter a decimal value when entering a quantity for the item at runtime. The Decimal Quantity is Allowed setting is set for each BOM item when you import a BOM Model, and it cannot be changed in Configurator Developer.

When importing a BOM Model into the CZ schema, the profile option CZ: Populate Decimal Quantity Flags controls whether an Oracle Configurator end user can enter a decimal quantity for BOM Standard Items that are defined as accepting decimal quantities in Oracle Bills of Material. For details about this profile option, see the *Oracle Configurator Installation Guide*.

Whether an item accepts decimal quantities or an integer also depends on the BOM Item Type of the item's parent Model. If the item's parent is an ATO BOM Model, the item was defined as accepting decimal quantities in Oracle Inventory, and the profile option CZ: Populate Decimal Quantity Flags is set to Yes, then the item accepts a decimal quantity in a runtime Oracle Configurator. (In this case, the Decimal Quantity is Allowed check box is selected in Configurator Developer.) If the item's parent is a PTO BOM Model, an end user cannot specify a decimal quantity for the item at runtime, regardless of the profile option's value, or how the item was defined in Oracle Inventory.

If an item accepts decimal quantities, an end user can enter up to 9 digits after the decimal character at runtime. For more information about importing BOM Models that use decimal quantities, see the *Oracle Configurator Implementation Guide*.

Warning: Oracle Bills of Material allows you to create BOM Models in which a divisible (decimal) parent has one or more indivisible (integer) children. However, Oracle Order Management does not allow users to order a BOM that is defined this way. Additionally, Oracle Configurator Developer displays an error when you generate logic for such a Model. For example, a BOM Option Class that allows decimal quantities must not contain Options that are defined as integers.

3.3.5.2 Decimal Quantities and Non-BOM Items

Features and Options accept either integers or decimal quantities at runtime. Totals and Resources can display up to two digits after a decimal point (for example, 3.12) while numeric Features display up to nine digits after a decimal (for example, 3.123456789).

3.3.6 Item Types and Imported BOM Properties

When you populate the CZ schema with data from Oracle Bills of Material, **Item Catalog Groups** defined in Oracle Inventory become **Item Types** in Configurator Developer. An Item Catalog Group is a related set of **Descriptive Elements** that are assigned to Inventory Items to provide additional information about an Item. Examples of Descriptive Elements include color, length, style, and weight.

The Descriptive Elements and **Descriptive Element Values** defined in an Item Catalog Group become each BOM item's Properties and Property Values, respectively, in Configurator Developer. This mapping is shown in Table 3–1 on page 3-5.

Table 3–1 Item Data Imported from Oracle Inventory

Oracle Inventory	Oracle Configurator Developer
Item Catalog Group Name	Item Type

Table 3–1 (Cont.) Item Data Imported from Oracle Inventory

Oracle Inventory	Oracle Configurator Developer
Descriptive Element	Item Property
Descriptive Element Value	Item Property Value

If no Item Catalog Groups are defined, each imported BOM item has an Item Type of Default Type. Items are children of Item Types. Therefore, an Item's parent indicates its type. For more information about the Item Master area of the Repository, see Chapter 26.

A profile option controls the default Item Type Name for each Item Catalog Group that is imported from Oracle Inventory. For more information, see the *Oracle Configurator Installation Guide*.

For more information about Item Catalog Groups and Descriptive Elements, see the *Oracle Inventory User's Guide*.

3.3.6.1 Data Types and Imported Items

Item Catalog Descriptive Element values do not have a data type in Oracle Inventory. When you import BOM Model data into the CZ schema, Descriptive Elements become Properties. Properties imported from Bills of Material have a data type of either Text or Decimal Number in Configurator Developer. The database setting `ResolvePropertyDataType` controls whether an imported Property's data type is set to Text or Decimal Number. This setting is described in the *Oracle Configurator Implementation Guide*.

For more information about data types, see Section 5.6, "Property Data Types" on page 5-11.

Because you can use Properties to define some types of configuration rules, it is important to consider whether you want Descriptive Element values to be imported as text or as numbers. It is also important to understand how changing a Descriptive Element's value in Oracle Inventory can have unintended results Configurator Developer.

For example, some of your configuration models contain Numeric Rules that use imported Properties. These Properties are derived from an Oracle Inventory Item Catalog Group called "Aluminum Pipe", which contains a Descriptive Element called "Length". All of the Descriptive Element values are numbers, such as 10, 15, 20, and so on. An Oracle Inventory user adds a Descriptive Element and, instead of just entering a number as its value, enters "20 *meters*". After refreshing any BOM Models that use the "Aluminum Pipe" Item Catalog Group, logic generation will fail for all of the Numeric Rules that use Properties from the Aluminum Pipe Item Type. All other configuration models that use this Item Type will also be affected.

3.3.6.2 Limitations when Modifying Imported Items, Item Types, and Properties

Because Oracle Bills of Material data must remain consistent in Configurator Developer, you cannot delete or modify imported Items, Item Types, or User Properties in the Main area of the Repository. For example, you cannot change the name or description of an imported Item or Item Type, or modify the value of an imported User Property, in the Main area of the Repository. However, you can modify a User Property when editing an Item or Item Type in the Item Master area of the Repository, with the following restrictions:

- You can add an imported or non-imported Properties when editing an Item Type, and modify Property values when editing an Item.
- If you or another Configurator Developer user assigned a Property to an Item Type in Configurator Developer, you can:
  - Remove the Property's association with the Item Type by editing the Item Type (this also removes the Property's association with all of the Item Type's child Items)
  - Modify the Property's value by editing the Item

  If a Property was imported with the Item, you cannot remove its association with the Item Type or modify its value.

You cannot export any Properties that you add to an Item Type to Oracle Bills of Material (that is, to update the BOM Model).

For information about modifying Items and Item Types, see Chapter 26. For additional information about Properties, see Chapter 5.

3.3.6.3 Limitation on BOM Model Structure and Item Effective Dates

In Oracle Bills of Material, you can define a BOM Model in which the same item (component) appears more than once beneath the same parent item, as long as the operation sequence numbers for each item are not the same. However, to configure such a Model in a runtime Oracle Configurator, the effective dates for the duplicate items must not overlap.

For example, in Oracle Bills of Material, Standard Item A appears multiple times as a component (child) of BOM Option Class X, and each instance of the item has a different operation sequence number. A runtime Oracle Configurator supports this structure only if the effective dates for each instance of Standard Item A do not overlap.

If the effective dates (or times) overlap at all, Oracle Configurator displays an error when an end user clicks Done to save the configuration. This is true even if the effective date defined in Oracle Bills of Material disables all but one instance of Standard Item A at runtime.

3.3.6.4 Advanced Product Catalog Item Attributes

Advanced Product Catalog (APC) is part of the Oracle Product Lifecycle Management application. If you have installed APC and use user-defined item attributes, you can enable business events for these attributes and import them into the CZ schema as item Properties when importing a BOM Model. To access the public API that maintains Descriptive Elements for a given item, see the Product Lifecycle Management documentation.

For more information, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.

3.3.7 Extending a BOM Model in Configurator Developer

You can include additional aspects of your configuration problem by adding structure to extend an imported BOM Model. For example, you might want to create Totals and Resources to keep track of a quantity or add nodes to present guided buying or selling questions to your Oracle Configurator end users. You can create nodes within a BOM Model, but not within a BOM Option Class or BOM Standard Item. When a BOM Model contains nodes created in Configurator Developer, the non-BOM nodes appear before the BOM nodes in the hierarchical structure (that is, as children of the BOM Model node itself).

For more information about building Model structure, see Chapter 29, "Structure Area of the Workbench" on page 29-1.

3.4 Container Models

A **Container Model** is a type of BOM Model that contains BOM Models, BOM Option Classes, and BOM Standard Items that are tracked in Oracle Install Base. This type of Model enables you to reconfigure installed instances of telecommunication services by moving, adding, changing, or disconnecting a customer's services in a runtime Oracle Configurator. Container Models are typically part of the Oracle Telecommunications Service Ordering (TSO) solution.

For more information about TSO, refer to the *Oracle Telecommunications Service Ordering Process Guide*.

4 References

This chapter presents information about Model References, such as how they are used in BOM Models, configuration rules, and User Interfaces.

This chapter includes the following sections:

4.1 Introduction

To reduce the time and effort required to create and maintain configuration models, a Model may contain one or more **References** to other Models. References allow a Model to be used as a subassembly within other Models. For example, your organization sells many different styles of trucks and automobiles, but some of them use the same 200 horsepower, V6 engine. You can create and maintain one Model for this engine in Oracle Configurator Developer, and then simply create a Reference to that Model from all other Models (automobiles) that use it. When the referenced Model is modified, the changes automatically propagate to all Models that refer to it.

Within the structure of a Model, a Reference node functions like a Component node. Like a Component, you can modify a Reference node's name and effectivity, and specify how many instances of the referenced Model are available and can be created in a configuration. (Instantiability is explained in Chapter 7.)

When you work in a Model that contains a Reference, the structure of the referenced Model appears as a subtree of the parent Model. All of the referenced Model's settings, structure, rules, and UIs are read-only when viewed from the parent Model.

At runtime, each instance contains the entire structure of the referenced Model and is subject to all the rules defined in that Model. Note that a Reference node functions differently in an imported BOM Model. For details, see Section 4.5, "References and BOM Models" on page 4-4.

4.2 References and Rules

Like Model structure, the rules in a referenced Model are read-only when you are working in the parent Model. However, you can use referenced Model nodes when defining configuration rules for the parent Model. All rules defined this way, even those whose participant nodes are all part of the referenced Model's structure, belong to and reside with the parent Model (that is, they do not belong to the referenced Model). You can create new rules for the parent Model, but all of the referenced Model's rules are read-only when viewed from the parent Model.

Note: Use caution when using nodes within a referenced Model as participants in the parent Model's configuration rules. If a node in the referenced Model is modified or deleted, the rule in the parent Model becomes invalid and Configurator Developer displays an error message when you generate logic for the parent.

If a Reference node can have multiple or variable instances, you can define a Numeric Rule that changes how many instances of the referenced Model can be created at runtime based on other end-user selections. This type of Numeric Rule is explained in Section 13.7 on page 13-4. Instantiability is explained in Chapter 7.

For information about viewing rules in a referenced Model, see Section 4.2 on page 4-2.

4.3 References and Effectivity

The root node of a Model is always effective. When a non-BOM Model is referenced by another Model, you can specify effective dates or assign an Effectivity Set, and specify one or more Usages to the Reference node. For References to BOM Models, you can modify only the Usage(s) assigned to the Reference node. To modify the effective dates of a "nested" referenced Model (that is, a Reference within a Reference), you must open its parent for editing. For example, Model 1 references Model 2, and Model 2 references Model 3. To modify the effective dates of Model 3, you must open Model 2 for editing in the Structure area of the Workbench. See Section 6.1, "Introduction" on page 6-1.

4.4 References and User Interfaces

At runtime, a UI is either invoked directly (when it is the UI for the item being configured) or it is invoked by reference (when it is the UI for a referenced Model). When you generate a UI, Configurator Developer selects the most recently generated or modified UI definition for each referenced Model. If a referenced Model has no UI, Configurator Developer generates a UI for it, then generates the parent Model's UI.

When a Model references other Models, the parent Model's UI provides controls to create and manage instances of each referenced Model (for example, a drilldown control for navigating to Pages within the referenced UI, controls for selecting options, and so on). For example, Model A references Model B. Model B is required and can have multiple instances at runtime. In the runtime UI, an end user can click a button to create, configure, or delete instances of Model B. The settings that control what types of controls are generated for referenced Models are described in

When a Model references another Model, the parent Model's UI also references the child Model's UI. If a referenced Model has multiple UIs, you can specify which one you want to use after generating the parent Model's UI. This procedure is explained in Section 31.3.1, "Modifying the User Interface Definition" on page 31-3.

4.4.1 Integrating Referenced User Interfaces

When configuring a referenced Model, the primary navigation style and images used to indicate selection state may be different than the parent Model's UI. For background, see Section 19.4, "Runtime Navigation" on page 19-10.

This section describes how parent and child (referenced) UIs with different primary navigation styles are integrated when you generate a UI, and what occurs at runtime when the end user transitions between parent and child UIs during a configuration session.

4.4.1.1 Non-Instantiable Child Model

When the child Model is not instantiable, its UI can be integrated with the parent UI in one of the following ways:

- It can be accessed via a drilldown link (hypertext, button, or image with a Go to Page action). This is the default for referenced UIs. This option is always available regardless of the combination of navigation mechanisms.
- It can be incorporated into the primary navigation mechanism of the parent model, that is linked from a side navigation Menu or dynamic Model Tree, accessed via step-by-step navigation, or presented as one or more subtabs. (The various primary navigation mechanisms are described in Section 20.2, "User Interface Master Templates" on page 20-2.)
- It can be incorporated into the "parent page" as a subsection, in its own Header region. This option is only applicable to a "single-page" UI. If this is the Master Template setting and a non-instantiable referenced UI is not a single page, the default option (drilldown) will be used instead, and Configurator Developer displays a warning message when you generate the parent Model's UI.

If the parent UI specifies Model Tree navigation and the child UI does not, the parent Model Tree contains a link to the initial page of the child UI. The internal structure of the child Model does not appear in the parent Model Tree. When the end user navigates to the child Model's initial page at runtime, its specified navigation mechanism will replace the Model Tree. When the end user navigates back to the parent Model, the Model Tree reappears.

If both the parent and child UIs specify Model Tree navigation, the parent Model Tree includes the entire Model Tree of the child at the appropriate location.

Integration of static navigation mechanisms, such as the single or multiple-level side Menus, occurs when you generate or refresh the parent Model's UI, rather than being integrated at runtime. This means you can edit the integrated contents of the side menu in the parent UI. (This integration only occurs for non-instantiable Model References.)

The integration of parent and child Model Tree structures occurs at runtime and the integration points are specified in the parent Model Tree. Configurator Developer sets the integration points when you generate or refresh the parent Model's UI, and bases them on the location of the Reference in the Model structure. You can modify the integration points when editing the contents of the parent Model Tree. (This is the same whether the referenced Model is instantiable or not.)

4.4.1.2 Instantiable Child Model

When the child Model is instantiable, the instances are always created and accessed via a UI element that provides access to instances at runtime (either an Instance Management Table or Single Instance Control Region, depending on the maximum number of instances permitted). In most cases, the child UI's primary navigation style becomes active, replacing the parent's navigation style.

If the parent UI specifies Model Tree navigation and the child UI does not, the parent Model Tree does not contain any child content. In this case, the end user navigates using the aforementioned instance control UI elements.

If both the parent and child UIs specify Model Tree navigation, the parent Model Tree will include the Model Tree of each existing child instance at the appropriate location.

Child instances must be created via instance control elements, UI Actions, or Configurator Extensions; the Model Tree does not provide another method of creating instances.

Integration of parent and child Model Tree structures occurs at runtime. Configurator Developer sets the integration points when you generate or refresh the parent Model's UI based on the location of the Reference in the Model structure. You can change the integration points when editing the contents of the parent Model Tree.

4.4.2 Modifying a Referenced User Interface

Configurator Developer stores a referenced Model's UI definition in the referenced Model, not in the parent Model. Therefore, a referenced Model's UI is read-only when viewed from its parent. To modify a referenced UI, you must first open the Model to which it belongs in the User Interface area of the Workbench.

You do not need to refresh the parent Model's UI after making changes to a referenced Model. Because the UI for a referenced Model is *linked* to the parent Model's UI, any changes to the referenced UI are reflected in the parent Model's UI automatically.

Configurator Developer prevents you from deleting a UI that is referenced by another Model, since doing so would cause pages to be missing from the parent Model's UI.

A referenced Model may have more than one UI. When modifying the parent Model's UI definition, you can select a different UI for any of its referenced Models. For details, see Section 31.3.1 on page 31-3.

4.4.3 Publishing and Referenced User Interfaces

If a UI Reference link is broken or missing, Configurator Developer displays a "consistency check" error when you publish the Model or update an existing publication. When this occurs, regenerate the parent Model's UI to recreate the links, and then republish the Model.

For details about republishing, see Section 23.4 on page 23-3.

4.5 References and BOM Models

One important use of References is to represent the relationship between a BOM Model that contains other BOM Models when you populate the CZ schema with BOM Model data. When you import a BOM Model that contains other BOM Models, Configurator Developer creates a Reference node for each child Model in the parent's structure. You can assign one or more Usages to a BOM Model Reference node and modify its instantiability settings, but all settings that are defined in Oracle Bills of Material are read-only in Configurator Developer (for example, the node's BOM Item Type, its Minimum, Maximum, and Default Quantity, and so on).

When a BOM Model is a child of (referenced by) several other BOM Models, it is not imported into Configurator Developer multiple times. The import procedure populates the CZ schema with the child BOM Model only once, and then creates References to it from each parent BOM Model. This allows the rules and UI for the referenced Model to be maintained in one place and ensures that no duplicate BOM Models exist in the database. For an example, see the *Oracle Configurator Implementation Guide*.

4.5.1 References and Optional BOM Models

A referenced BOM Model is not required to create a valid configuration if the Optional check box is selected for that Model in Oracle Bills of Material. If an optional BOM Model contains a Feature that is required, unintended results may occur at runtime. A required Feature is any Option Feature that has a Minimum greater than 0 (zero), or a Text Feature that has the Required check box selected in Configurator Developer.

This can occur at runtime when an optional BOM Model has a required Feature:

- At the Model level
- In a child (referenced) Model

  This applies whether the Reference is to an imported BOM Model or a non-imported Model.

Example: Model A references Model B and Model C, which are both optional BOM Models. Model B and C are variations of the same component, and a valid configuration may contain one and only one of these components. Both Models contain a required Feature, and reference a Model that contains a required Feature (this structure is shown in Figure 4–1 on page 4-6).

At runtime, the end user selects and configures Model B, satisfying the configuration requirement for that component. However, a selection from Model C is still required because it contains a required Feature. In this situation, the end user will be able to save a valid configuration of Model A, since only Model B *or* Model C is allowed. However, the configuration will still be incomplete.

To avoid this situation, do the following:

- Make any required Features optional by changing the Minimum Selections to 0.
- If the end user must make a selection from the Features to create a valid configuration, create a Logic Rule that uses the Requires relation to tie the Features to the parent BOM Model.

  For example, using the Model shown in Figure 4–1 on page 4-6, change the Initial Minimum for all of the required Features to 0, and then create the following rules:

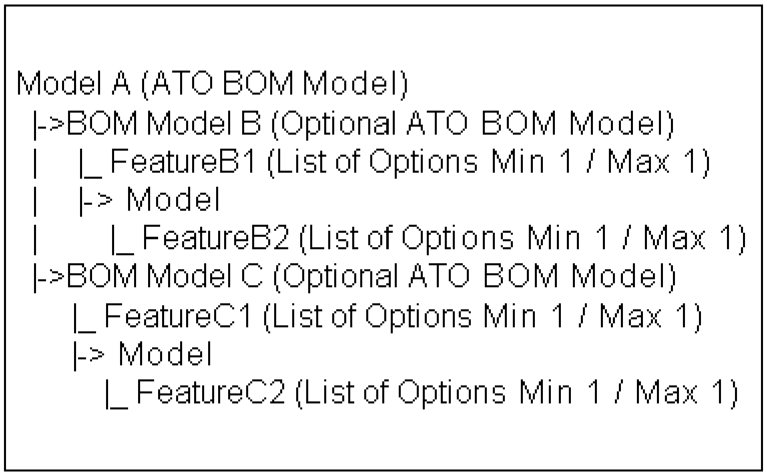
  "BOM Model B REQUIRES FeatureB1"

  "BOM Model B REQUIRES FeatureB2"

  "BOM Model C REQUIRES FeatureC1"

  "BOM Model C REQUIRES FeatureC2"

Figure 4–1 Optional BOM Models with Required Features



```
Model A (ATO BOM Model)
 |->BOM Model B (Optional ATO BOM Model)
 |     |_ FeatureB1 (List of Options Min 1 / Max 1)
 |     |-> Model
 |          |_ FeatureB2 (List of Options Min 1 / Max 1)
 |->BOM Model C (Optional ATO BOM Model)
       |_ FeatureC1 (List of Options Min 1 / Max 1)
       |-> Model
            |_ FeatureC2 (List of Options Min 1 / Max 1)
```

4.5.2 Creating Model References in Configurator Developer

You can create a Reference from a non-imported Model to one BOM Model and multiple non-imported Models. However, you cannot create a Reference from a BOM Model to another BOM Model. Although an imported BOM Model can contain one or more other ATO or PTO Models, this type of structure can be created and maintained only in Oracle Bills of Material. This is because all imported BOM Models must retain their structure as defined in Oracle Bills of Material. If a BOM Model's structure could be modified in Configurator Developer (by referencing another BOM Model, for example), the resulting Model would not be orderable from Oracle Order Management.

Additionally, you cannot create a Reference in a BOM Model to a non-imported Model that references a BOM Model. For example, a non-imported Model called M1 references BOM Model B1. You cannot create a Reference from another BOM Model to M1, since this would create Model structure in which a BOM Model references another BOM Model. As mentioned above, this type of structure can only be created in Oracle Bills of Material.

Creating a Model Reference is explained in Section 29.8 on page 29-5.

Note: Running the Refresh All Configuration Models concurrent program may cause additional Models and References to be created if, for example, new structure has been added to the root BOM Model in Oracle Bills of Material.

4.6 Updating Referenced Models

When a Model is the target of a Reference and you make any changes to the structure of the referenced Model, you may need to update the parent Model. Updating may include generating logic, refreshing UIs, and republishing the parent and child Models (if they were previously published). This is because:

- Rules in the parent may refer to deleted structure nodes in the child.
- Newly added Options in the child may be missing from Explicit Compatibility Rules in the parent.

The logic generated for Property-based Compatibility Rules in the parent Model that refer to Features in the child Model may change when you modify Properties in the child (referenced) Model. Property-based Compatibility Rules are described in Section 15.2.2 on page 15-3.

Configurator Developer does not automatically update the parent Model when you modify the structure of a referenced Model. You may be able to resolve any potential problems with the configuration model simply by regenerating logic for the parent Model. When you generate logic for a Model, Configurator Developer generates logic for any of its referenced Models as necessary.

Not all changes that you make to a Model that is referenced require you to update its parent Model(s). For example, modifications to the configuration rules or UI definitions of a referenced Model do not require changes within the parent, except perhaps for re-testing.

You cannot delete a Model if it is referenced by another Model.

For information about limitations that exist when refreshing a BOM Model that contains other BOM Models, see the *Oracle Configurator Implementation Guide*.

4.7 Copying Models with References

When copying a Model that contains References, you can either create a new copy of each referenced Model, or maintain the existing Model References. This is an important consideration because changes to a Model affect every other Model that references it.

Select one of the following when copying a Model with References:

- **Maintain Existing References**: Select this option if you want both the original Model and the copy of that Model to refer to the *same* Models. In this case, any changes made to a referenced Model affect the original Model and the copied Model.

  For example, Model A contains references to B1 and B2. You copy Model A and rename it New Test Model. Both Model A and New Test Model now refer to Model B1 and Model B2. Therefore, any changes made to B1 or B2 will affect both Model A and New Test Model.

- **Copy Entire Reference Chain**: Select this option if you want to create copies of all referenced Models.

  For example, Model A contains references to B1 and B2. You copy Model A and rename it New Test Model. Model A still references Model B1 and Model B2, but New Test Model references "Copy (1) of B1" and "Copy (1) of B2". Therefore, changes to B1 and B2 affect only Model A, while changes to Copy (1) of B1 or Copy (1) of B2 affect only New Test Model.

WARNING: Copying an imported BOM Model and selecting Copy Entire Reference Chain can cause an error when you publish one of copied Models. Configurator Developer displays a warning message when you publish a copy of a BOM Model if any of the following are true:

- **The original BOM Model was published previously**
- **The original Model and the copy have the same Product Key**
- **The new publication and the existing publication have overlapping effectivity dates**

You can view a Model's Product Key in the General area of the Workbench.

Note: You can also copy a Model and all of its child (referenced) Models programmatically using the PL/SQL package `CZ_modelOperations_pub`. This package contains a set of APIs that automate many of the tasks required to maintain configuration models. For more information, see the *Oracle Configurator Implementation Guide*.

4.8 Editing a Model Reference Node

You can perform the following operations on a Reference node in Configurator Developer:

- Rename it (see Section 25.5.3, "Renaming Objects and Modifying Descriptions" on page 25-11)
- Change its effective dates and Usages (see Section 4.3, "References and Effectivity" on page 4-2)
- Delete it: This disassociates the referenced Model, its rules, and UIs from the parent Model. This action does not affect the referenced Model itself in any way.
- Modify its Instantiability settings.

  To modify the Initial Minimum and Initial Maximum values for a Reference node, the node's parent Model must be open for editing in the Structure area of the Workbench. For more information about these settings, see Section 7.1, "Introduction" on page 7-1.

When editing a Model that contains one or more References to other Models, you *cannot* modify or delete the referenced Model's structure, rules or User Interface(s).

5 Properties

This chapter describes Properties, their relationship to Model structure, and how they can be used when defining rules and runtime conditions.

This chapter includes the following sections:

- User Properties
- System Properties
- Configuration Session Properties
- User Properties on Structure Nodes and Items
- Property Data Types

5.1 Introduction

All Model structure nodes have attributes called Properties. An end user cannot select or modify Properties in a runtime Oracle Configurator, but Properties and their values can be used when defining rules or to create runtime UI captions for Model structure nodes.

For more information, see:

- Chapter 15, "Comparison and Compatibility Rules"
- Section 13.7, "Using Properties when Defining a Numeric Rule" on page 13-4.
- Section 28.9, "Runtime Display Names" on page 28-6

There are two types of Properties: User Properties and System Properties.

5.2 User Properties

You can create User Properties in Configurator Developer, or they may be created in Oracle Bills of Material and added to the CZ schema when importing a BOM Model. Examples of User Properties include Size, Weight, Length, Diameter, and Color.

User Properties can also be modified or deleted, but Properties created in Oracle Bills of Material can only be modified or deleted in that application. The same is also true for User Properties created in Configurator Developer. You create and manage User Properties that were created in Configurator Developer in the Main area of the Repository.

For details about User Properties that are imported with a BOM Model, see Section 3.3.6, "Item Types and Imported BOM Properties" on page 3-5.

You can add User Properties to Item Types in the Item Master area of the Repository, and to Model nodes when working in the Structure area of the Workbench. You can also modify Property values when working on these objects, with some restrictions. For details, see:

- Section 3.3.6.2, "Limitations when Modifying Imported Items, Item Types, and Properties" on page 3-6
- Section 5.5, "User Properties on Structure Nodes and Items" on page 5-10

5.3 System Properties

System Properties are implicit attributes of both BOM and non-BOM nodes that vary based on the node's type. Examples of System Properties include Name, Description, Quantity, and Value.

You can use System Properties when defining:

- Logic, Numeric, and Comparison Rules

  For more information, see Appendix C, "Rules, Node Types, and System Properties" on page C-1.
- The source for a UI element's caption

  See Section 21.12, "User Interface Element Captions and Details" on page 21-35.
- A runtime condition

  See Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

When you select a Property from a list in Configurator Developer (for example, when defining a rule), an open and closed parenthesis is appended to each System Property.

For example:

```
Name()
Description()
MinInstances()
MaxInstances()
Weight
Length
Diameter
```

In this example, `Name`, `Description`, `MinInstances`, and `MaxInstances` are System Properties, while `Weight`, `Length`, and `Diameter` are User Properties.

Table 5–1 on page 5-2 lists all System Properties and indicates how each can be used.

Table 5–1 System Properties

System Property	Description	Node Type(s)	Returns	Use
DisplayName	Default UI caption for Model structure nodes (specified in the General area of the Workbench)	All nodes	Text	Caption text and runtime conditions
DisplayNamePath	Path of default node UI captions from the Model root	All nodes	Text	Caption text and runtime conditions

Table 5–1 (Cont.) System Properties

System Property	Description	Node Type(s)	Returns	Use
Name	Node name	All nodes	Text	Caption text and runtime conditions
Description	Node description	All nodes	Text	Caption text and runtime conditions
InstanceName	Instance name	Instantiable Model References and Components	Text	Caption text and runtime conditions
InstanceNumber	Instance number	Instantiable Model References and Components	Integer	Caption text and runtime conditions
InstanceCount	Number of runtime instances	Instantiable Model References and Components	Integer	Rules, caption text, and runtime conditions
Quantity	Node quantity	BOM Option Classes and BOM Standard Items, Options, Count Features	Integer	Rules, Caption text, and runtime conditions
BaselineQuantity	Quantity restored from Oracle Install Base Used only when reconfiguring installed instances. For more information, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	Integer	Caption text and runtime conditions
DeltaQuantity	Difference between current quantity and quantity restored from Oracle Install Base. Used only when reconfiguring installed instances. For more information, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	Integer	Caption text and runtime conditions

Table 5–1 (Cont.) System Properties

System Property	Description	Node Type(s)	Returns	Use
Value	Integer value	Totals, Resources, Text Features, Integer Features, Decimal Features, Text Features	Decimal, for Totals, Resources, and Decimal Features Integer for Integer Features Text for Text Features True or False for Boolean Features	Can be used for caption text and runtime conditions (regardless of node type). When node type is Decimal Feature, Integer Feature, or Boolean Feature, can be used only in rules.
SelectionState	An option's runtime selection state (Selected, Selectable, or Excluded) See Section 5.3.1, "Selection State" on page 5-7.	BOM Models, BOM Option Classes, Option Features that have Maximum Selections set to 1	StateQuantity Node	Caption text and runtime conditions
DetailedSelectionState	An option's detailed runtime selection state For details, see Section 5.3.1, "Selection State" on page 5-7	All BOM nodes, all types of Features, and Options	User Selected, User Declined, System Selected, System Excluded, or Selectable	Caption text and runtime conditions
LogicState	Node logic state (User True, Logic True, and so on)	All BOM nodes, Options, Option Features, Boolean Features, Count Features	User True, User False, Logic True, Logic False, Unknown	Rules
NodeUnsatisfied	Returns True when the node is unsatisfied (that is, a selection or input is required) For more information, see Section 11.5, "Configuration Rules and Logic State" on page 11-5.	All nodes	True or False	Caption text and runtime conditions

Table 5–1 (Cont.) System Properties

System Property	Description	Node Type(s)	Returns	Use
SubtreeUnsatisfied	Returns True when the node or any of its descendants is unsatisfied (that is, a selection or input is required)	All nodes	True or False	Caption text and runtime conditions
Valid	Is the value entered for the node valid (that is, within the specified range)	All nodes	True or False	Caption text and runtime conditions
ListPrice	Item list price	All BOM nodes	Decimal	Caption text and runtime conditions
SellingPrice	Item selling price	All BOM nodes	Decimal	Caption text and runtime conditions
ExtendedPrice	Quantity multiplied by the item selling price	All BOM nodes	Decimal	Caption text and runtime conditions
ATP	Available-to-promise date	All BOM nodes	Date	Caption text and runtime conditions
MinQuantity	Minimum quantity	All BOM nodes, Count Features	Decimal	Caption text and runtime conditions
MaxQuantity	Maximum quantity	All BOM nodes, Count Features	Decimal	Caption text and runtime conditions
MinValue	Maximum integer or decimal value	Integer Features, Decimal Features	Integer	Caption text and runtime conditions
MaxValue	Maximum integer or decimal value	Integer Features, Decimal Features	Integer	Caption text and runtime conditions
MinSelected	Minimum number of options selected	BOM Models, BOM Option Classes, Option Features	Integer	Caption text and runtime conditions
MaxSelected	Maximum number of options selected	BOM Models, BOM Option Classes, Option Features	Integer	Caption text and runtime conditions
MinInstances	Minimum instances	Instantiable Model References and Components	Integer	Rules, Caption text, and runtime conditions

Table 5–1 (Cont.) System Properties

System Property	Description	Node Type(s)	Returns	Use
MaxInstances	Maximum instances	Instantiable Model References and Components	Integer	Rules, Caption text, and runtime conditions
HasChildren	Node has children	All nodes	True or False	Caption text and runtime conditions
TargetDisplayName	Display name of target instance	Connectors	Text	Caption text and runtime conditions
TargetDisplayNamePath	Display name path of target instance	Connectors	Text	Caption text and runtime conditions
LineType	The line type as set by a Functional Companion or Configurator Extension Rule Used only when reconfiguring installed instances. For details, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	Text	Caption text and runtime conditions
Location	Item location as set by a Configurator Extension Rule Used only when reconfiguring installed instances. For details, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	Integer	Caption text and runtime conditions
IBNodeChanged	Item modified relative its state in Oracle Install Base. Used only when reconfiguring installed instances. For details, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	True or False	Caption text and runtime conditions

Table 5–1 (Cont.) System Properties

System Property	Description	Node Type(s)	Returns	Use
IBSubtreeChanged	True if item or one of its descendants has changed relative to Oracle Install Base. Used only when reconfiguring installed instances. For details, see the *Oracle Telecommunications Service Ordering Process Guide*.	All BOM nodes	True or False	Caption text and runtime conditions
Options	Children (options) of parent node	BOM Models, BOM Option Classes, Option Features	StateQuantity Node	Rules
Parent	Parent node	All nodes	Node	Rules
Selection	Currently selected node	BOM Models, BOM Option Classes,	Node	Rules
Children	Returns a collection containing the current node’s children	All nodes	Collection of iRuntimenode	Configurator Extension argument bindings
SelectableChildren	Node has selectable children	Component, BOM Models, BOM Option Classes, Option Features	Option (yes/no)	Rules
SummaryChildren	Returns a collection containing the child nodes of the current node that can be displayed in the Configuration Summary page	All nodes	Collection of iRuntime node	Configurator Extension argument bindings
Target	Target instance of the connection	Connectors	Component	Configurator Extension argument bindings
Eligible Targets	Eligible target instances of the connection	Connectors	Collection of components	Configurator Extension argument bindings

5.3.1 Selection State

Before reading this section, you should understand the term "logic state." For details, see Section 11.5, "Configuration Rules and Logic State" on page 11-5.

The `DetailedSelectionState` System Property refers to an option's runtime selection state. A runtime Oracle Configurator checks an option's `DetailedSelectionState` System Property to determine how to display its status. Depending on the value of this System Property, the UI Master Template you used to generate the UI determines which indicator or control image to display (see Section 20.2.2.7, "Images Section" on page 20-9). For example, you may want to display a status icon next to a standard HTML check box or radio button, or use an Enhanced Check Box or Enhanced Radio Button to indicate each option's selection state. An option's current `DetailedSelectionState` value also appears in the runtime UI when an end user places the cursor over an indicator or control image.

The `SelectionState` System Property is a simplification of `DetailedSelectionState`. It has only three values: Selected, Selectable, and Excluded. Therefore, SelectionState is not used to indicate whether an option's state is set by a user's action or by a configuration rule. This System Property is useful primarily for specifying runtime conditions, in which how an option is selected or excluded is less important than whether to hide or disable a UI element. See Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

Table 5–2 lists each logic state and their corresponding detailed selection states.

Table 5–2 Logic State, Selection State, and Detailed Selection State Values

Logic State	Selection State	Detailed Selection State
Unknown	Selectable	Not Selected
User True	Selected	Selected
Logic True	Selected	Auto-Selected
Logic False	Excluded	Auto-Excluded
User False	Selectable	Declined

The Not Selected state indicates that no decision has yet been made about an option, and it can be selected. An option has the Selected state when an end user selects it. Options selected by the propagation of a rule have a status of Auto-Selected, while options that are excluded from the configuration by a rule are Auto-Excluded. An option has the Declined state when an end user deselects an option that was previously selected by a rule (that is, its logic state was Logic True).

5.3.1.1 Displaying Logic False Options

In some situations, the runtime Oracle Configurator displays options that have a logic state of Logic False (Auto-Excluded) as Unknown (Not Selected).

For example, a BOM Option Class contains 5 mutually exclusive options (BOM Standard Items). When an end user selects an option, the remaining options become Logic False, but the runtime UI displays them as Unknown (that is, using the image specified in your UI Master Template for "Not Selected"). This behavior is considered preferable to displaying Logic False options as "excluded", because an end user might incorrectly assume that an excluded option cannot be selected.

5.4 Configuration Session Properties

Configuration session properties are distinct from User and System Properties because they refer to the state of a runtime configuration session, and are not associated with Model structure nodes.

You can use a configuration session property when defining:

- The source for a UI element's caption

  See Section 21.12, "User Interface Element Captions and Details" on page 21-35.

- A runtime condition

  See Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

Table 5–3 describes each configuration session property and indicates how each can be used.

Table 5–3 Configuration Session Properties

Property	Description	Returns	Use
ModelQuantity	Model quantity	Integer	Caption text and runtime conditions
CurrencyCode	Currency	Text (currency string, such as USD)	Caption text and runtime conditions
TotalListPrice	Total list price for all selected items	Decimal	Caption text and runtime conditions
TotalSellingPrice	Total selling price for all selected items	Decimal	Caption text and runtime conditions
ATPRollup	ATP Date for the entire configuration	Text (date format)	Caption text and runtime conditions
Valid	The configuration is valid	True or False	Caption text and runtime conditions
Unsatisfied	Returns true when the configuration is unsatisfied (that is, a selection or input is still required)	True or False	Caption text and runtime conditions
PricingEnabled	The configuration session is set up to display list prices or selling prices.	True or False	Caption text and runtime conditions
ListPriceEnabled	The configuration session is set up to display item list prices.	True or False	Caption text and runtime conditions
SellingPriceEnabled	The configuration session is set up to display item selling prices.	True or False	Caption text and runtime conditions
ATPEnabled	The configuration session is set up to display Available to Promise (ATP) information for selected items.	True or False	Caption text and runtime conditions

Table 5–3 (Cont.) Configuration Session Properties

Property	Description	Returns	Use
PriceAndATPDisabled	The configuration session is not set up to display item prices or ATP data.	True or False	Caption text and runtime conditions
InNestedTransaction	Returns true if the session is currently in a nested transaction. See Section 20.2.2.2.1, "Defining Custom Pagination and Layout" on page 20-5.	True or False	Caption text and runtime conditions
IsContainerModel	Returns true if configuring a Model that has Connectors. For more information, see the *Oracle Telecommunications Service Ordering Process Guide*.	True or False	Caption text and runtime conditions
HasInstalledRevisions	Returns true if the configuration has one or more instances with installed revisions. For more information, see the *Oracle Telecommunications Service Ordering Process Guide*.	True or False	Caption text and runtime conditions
ConfigHeaderID	Configuration header ID number	Integer	Caption text and runtime conditions
ConfigRevisionNumber	Configuration revision number	Integer	Caption text and runtime conditions

5.5 User Properties on Structure Nodes and Items

For the portions of your Model that you create in Oracle Configurator Developer, you can add, remove, and edit User Properties directly on nodes of the Model, or add User Properties to Items in the Item Master. Adding or deleting a User Property assigned to a Model node affects only the selected node. In other words, User Properties that you add directly to a Model's structure do not affect the Item Master. However, adding or removing User Properties in the Item Master affects all Items of the selected Item Type.

For example, when you add a Property to an Item Type, Configurator Developer automatically assigns the Property to all of its Items. If a Model contains structure created from the Item Type, repopulating the Model adds the Property to all of the nodes originally created by the Populator.

For more information, see:

- Section 2.1, "Introduction" on page 2-1
- Section 3.3.6, "Item Types and Imported BOM Properties" on page 3-5
- Chapter 10, "Using Populators"

In Configurator Developer, you can assign User Properties to imported nodes, including BOM Models, BOM Option Classes, and BOM Standard Items. This is described in Section 29.14, "Adding Properties to a Model Node" on page 29-9.

However, you cannot add Properties to a node that is a Reference to a BOM Model or to any of the nodes within the referenced structure.

When modifying an imported BOM node, you can edit or remove any User Properties that were added to the node in Configurator Developer. However, all User Properties imported with a node when you imported the BOM Model are read-only and cannot be modified or deleted in Configurator Developer. Imported Properties can be modified or deleted only in Oracle Bills of Material, and you must refresh the BOM Model to see the changes in Configurator Developer.

The Imported column in a BOM Model node's details page indicates whether a Property was imported with the node. In other words, it does not contain a check mark if the Property was imported from Oracle Bills of Material, but was not assigned to the node at the time. For more information, see Section 3.3.6.2, "Limitations when Modifying Imported Items, Item Types, and Properties" on page 3-6.

You can create Model nodes using Items in the CZ schema's Item Master by selecting "Item-based Nodes" in the Create Nodes page, or by running a Populator. When you create nodes using either method, the node always reflects all of the Properties and Property values of the source node in the Item Master. Properties assigned to a node this way are known as inherited Properties and are indicated by a column in a Model node's details page.

For more information about creating Model structure from Items or Item Types, see:

- Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4
- Chapter 10, "Using Populators"

For more information on working with Properties, see:

- Section 25.3.3, "Creating a Property" on page 25-3
- Section 25.4.2, "Modifying Properties" on page 25-6
- Section 26.6, "Adding Properties to Items and Item Types" on page 26-3
- Section 26.5, "Editing Properties Assigned to Items or Item Types" on page 26-2

5.6 Property Data Types

All User Properties have a data type that cannot be changed in Configurator Developer, regardless of whether the Property was imported or created in Configurator Developer. The data type determines what you can enter for a Property's value (for non-imported Properties) and how the Property can be used in a configuration rule. For example, a Property-based Compatibility Rule defines an expression that compares the User Property values of two nodes. Combinations of Properties that render the expression 'true' are considered compatible.

User Properties can also be imported from Oracle Bills of Material. For more information, see Section 3.3.6.1, "Data Types and Imported Items" on page 3-6.

Following are the available User Property data types:

- Integer (for example, 21)
- Decimal Number (for example, 11.34)
- True/False

- Text: For example: Property Name = Size, Property Value = Large. You may want to use this type if, for example, you use Property values to generate the default UI captions for Model structure nodes. For details, see Section 28.9, "Runtime Display Names" on page 28-6.

  User Properties with a data type of Text can be used only in Comparison Rules.

- Translatable Text: Select this type if you plan to implement Multiple Language Support (MLS) in the runtime Oracle Configurator and intend to display the Property's value in the UI. For more information about MLS, see Appendix B.

  Using Properties with this data type in rules is not recommended, as the rule may become invalid when the text (the Property's value) is translated.

6 Effectivity

This chapter describes how to use effectivity when building configuration models and defining rules.

This chapter includes the following sections:

6.1 Introduction

Effectivity allows you to model a product that changes over time. The effectivity you assign to Model nodes and rules determines whether a node is available or a rule is active in a runtime Oracle Configurator, or when unit testing a configuration model from Configurator Developer. Oracle Configurator and Oracle Configurator Developer use the database date and time when evaluating an object's effectivity.

You can also use the Effectivity Date Filter setting to control whether ineffective Model structure nodes and rules are displayed when working in Configurator Developer. For details, see Section 24.3.3.2, "Effectivity Date Filter" on page 24-9.

You control effectivity for a node or rule by indicating that it is either Always Effective or Never Effective, or by specifying either a date range or an **Effectivity Set**. You can also optionally assign one or more **Usages** to control effectivity of structure nodes and rules. Usages are described in Section 6.4 on page 6-2.

Defining effectivity for a node is explained in Section 29.17.12 on page 29-14. Defining effectivity for a rule is explained in Section 30.19.3 on page 30-19.

Effectivity also controls the availability of a Model publication to host applications. The application that is hosting the runtime Oracle Configurator specifies the date, time, and a Usage in an **initialization message**. All nodes that are not effective when the Model is invoked do not appear in the runtime UI. Similarly, all configuration rules that are not effective are ignored. For details about the initialization message, see the *Oracle Configurator Implementation Guide*.

For details about effectivity and publishing, see Section 23.5, "Applicability Parameters" on page 23-4.

You can also specify effectivity when unit testing a configuration model using the Model Debugger or a runtime User Interface. For details, see Chapter 22, "Testing and Debugging".

The root node of a Model is always effective, and its Effectivity settings are read-only. The Effectivity settings on a BOM Model References are also read-only because they are imported with the BOM Model. However, you can modify the effectivity on a Reference to a non-imported Model. For more information about Model References, see Chapter 4.

Note: When you populate the CZ schema with BOM data, the effective dates defined for each BOM item in Oracle Bills of Material are also imported, but like other imported data, are read-only in Configurator Developer. Additionally, you cannot assign an Effectivity Set or specify a Usage exception to imported BOM items.

6.2 Date Ranges

The effectivity for imported BOM nodes is defined as a date range in Oracle Bills of Material, and cannot be changed in Configurator Developer.

You can specify an effective date range for a rule or any node created in Configurator Developer. A date range may include both a start date and an end date (day, month and year) and time (hours, minutes, and AM or PM), or you can select either No Start Date or No End Date. (Selecting both No Start Date and No End Date is the same as selecting Always Effective).

To specify an effective date range for a node or rule, see Section 29.17.12 on page 29-14.

6.3 Effectivity Sets

Create an Effectivity Set to define an effectivity date range that can be shared by many Model structure nodes and configuration rules simultaneously. When you modify an Effectivity Set’s date range, the change affects all nodes and rules that are assigned to the Effectivity Set. Therefore, if you expect to use a specific effectivity date range for more than a very limited number of nodes, it is better to define and assign an Effectivity Set rather than specifying an effective date range for each node and rule in your Model.

You can assign Effectivity Sets to rules, Components, Features, Options, Totals, Resources, Model publications and References to non-imported Models. The effectivity assigned to imported BOM nodes is read-only in Configurator Developer, and can be changed only in Oracle Bills of Material.

An Effectivity Set can be always effective, never effective, or effective only within the range of dates that you specify.

You can also assign an Effectivity Set to rules that are part of a Rule Sequence. See Section 18.4, "Rule Sequences and Effectivity Sets" on page 18-2.

See Section 25.3.4, "Creating an Effectivity Set" on page 25-4.

6.4 Usages

Like effective dates and Effectivity Sets, Usages provide a method of controlling the effectivity of Model structure, rules, and the availability of Model publications to a host application. A host application may pass a Usage as a parameter in its initialization message, but it is not required. You can assign Usages independently or in addition to date effectivity (in other words, either an explicit date range or an Effectivity Set).

Usages consist of any text string that you specify, and you create them in the Main area of the Repository. You can create a maximum of 64 Usages.

By default, all Model structure nodes and any rules that you define are effective for all Usages. Setting a Usage on a node or rule means that it is effective for all Usages *except* the Usage(s) that you specify.

For example, Component A and Logic Rule X are effective for all Usages except the Usage called "Experienced User." At runtime, Oracle Order Management specifies this Usage in the initialization message. As a result, Component A does not appear in the UI and Logic Rule X is ignored.

Usages are a powerful tool for manipulating the effectivity of Model structure, rules, and Model publications based on a variety of business requirements. Therefore, be sure to plan for and implement Usages very carefully when developing configuration models. For example, it is not advisable to assign a Usage on a Reference to a required BOM Model because the item will not be available when the host application passes the Usage at runtime.

For an example of how you can use Usages to limit the effectivity of Model structure, rules, and Model publications, see the *Oracle Configurator Implementation Guide*.

See Section 25.3.5, "Creating a Usage" on page 25-4.

To make a node ineffective for a specific Usage, see Section 29.17.12 on page 29-14.

To make a rule ineffective for a specific Usage, see Section 30.19.3 on page 30-19.

6.5 Filtering Model Nodes and Rules in Configurator Developer

Effectivity is typically used when unit testing a configuration model in a generated UI or the Model Debugger, or in a deployed Oracle Configurator in a production environment. However, you can also hide ineffective Model structure nodes and rules when building a configuration model in Configurator Developer.

The Effectivity Date Filter setting in the Configurator Developer Preferences page controls whether Configurator Developer considers effectivity when displaying Model structure nodes and rules. For more information, see Section 24.3.3.2, "Effectivity Date Filter" on page 24-9.

6.6 Effectivity Examples

Following are some examples of how effectivity affects a configuration model in a runtime Oracle Configurator or when unit testing from Configurator Developer:

- Model structure nodes that are not effective do not appear, and therefore are not available for selection.

  For example, a specific item in the Model will be obsolete as of May 1, 2004. You want to automatically discontinue that item as of that date so it will not be offered as an option to your end users. By specifying an effective end date of 05/01/2004 for the item, it no longer appears in the configuration model on that date or in any future configuration sessions.

- Configuration rules that are not effective are ignored (do not propagate) at runtime.

For example, your configuration model contains a Logic Rule that selects several options based on a guided buying or selling question. However, several of these options will not be available (are no longer effective) after the end of the fiscal year. You define effectivity for the rule so it is ignored as of the date you specify. This prevents your end users from seeing any unnecessary warnings or error messages that may appear because some items are unavailable.

- Rules that contain ineffective participants may be unable to perform their intended actions at runtime.

  For example, your Model is an air compressor which includes a compression gauge and feeder hose as selectable options. You define a Logic Rule that automatically selects a compression gauge when the end user selects a feeder hose. However, when an end user configures the Model, the compression gauge is unavailable due to its effectivity. In this case, the rule is triggered when the end user selects the feeder hose, but Oracle Configurator displays a message informing the end user that the compression gauge is not available.

For an example of how to implement Usages in a configuration model, see the *Oracle Configurator Implementation Guide*.

7 Instantiation

This chapter describes how to build a configuration model that allows Oracle Configurator end users to create multiple instances of configurable components at runtime.

This chapter includes the following topics:

- Multiple Instantiation Conditions
- Modifying Instantiability
- Loading Models with Instantiable Components
- Runtime Display of Instantiable Components

7.1 Introduction

Instantiability refers to an end user's ability to create and individually configure one or multiple occurrences (instances) of a Model or Component in a runtime Oracle Configurator. A Model Reference or Component node's Instantiability settings determine how many instances of the component exist when the configuration session begins, and whether an end user can create additional instances at runtime. These settings are described in Section 29.17.6 on page 29-13.

At runtime, the end user accesses an instance of a configuration model, as well as an instance of each component contained within the Model. The end user configures each component instance separately by selecting from available options within that component.

Note: This user's guide uses the word "component" when referring to an instance of a Model or a Component node in a runtime Oracle Configurator.

For example, a computer system can be represented by a Model. A computer may contain a number of different servers, printers, and personal computer (PC) workstations, all of which are child Models of the computer system Model. Each PC workstation represents one instance of a child Model, and each instance of the PC workstation can be configured differently.

To continue the example, one instance of the PC workstation can be configured with a 21 inch flat screen monitor, 10GB of disk space and 512 KB RAM, whereas another instance of the PC workstation has a 17 inch screen, ergonomic keyboard, 256 KB RAM and 4 GB of disk space. These two workstations are part of the computer system Model.

Note: The Generic Configurator UI does not support multiple instantiation. An Oracle Configurator end user can create multiple instances of components only in User Interfaces that you create in Configurator Developer. For more information about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

7.2 Multiple Instantiation Conditions

A node's instantiability does not affect publishing, batch validation, or saving and restoring configurations.

7.2.1 Importing and Refreshing BOM Models

To allow child BOM Models to be instantiated multiple times in a runtime Oracle Configurator, the root node of an imported BOM Model be either a PTO (Pick-to-Order) or an ATO (Assemble-to-Order) BOM Model. You specify the Model Type when defining the BOM Model in Oracle Bills of Material. See Section 7.2.2, "Nodes that Are Instantiable" on page 7-2 for more information.

Refreshing an imported BOM Model updates the default Quantity specified in Oracle Bills of Material, but does not update the values of each node's Instantiability settings.

For more information about importing BOM Models, see the *Oracle Configurator Implementation Guide*.

7.2.2 Nodes that Are Instantiable

Configurable components include Components and Model References. In Configurator Developer, the Instantiability setting is available for all configurable components under the root node. If a configurable component's Instantiability setting is Multiple or Variable Instances, then an end user can create at least one instance of the component at runtime.

For details about modifying a node's instantiability, see Section 7.4 on page 7-3.

The following nodes are instantiable:

- A Reference to a PTO BOM Model that is a child of another PTO BOM Model
- A Reference to an ATO BOM Model that is a child of another ATO BOM Model
- A Reference to an ATO BOM Model that is a child of a PTO BOM Model
- A Reference to a Model created in Configurator Developer
- A Component

The first time you import a BOM Model, Configurator Developer sets the Instantiability setting to Required Single Instance. If you then modify the Instantiability settings on any BOM Model References, Configurator Developer does not update them when you refresh the BOM Model.

Note: The BOM Item Type field in Configurator Developer indicates whether a BOM item is an ATO Model, PTO Model, Option Class, or Standard Item.

7.2.3 Nodes that Are Not Instantiable

The following nodes are not instantiable:

- The top-level (root) Model node
- A BOM Model that is a child of a non-imported Model
- BOM Option Classes, BOM Standard Items, Features, Options, Totals, Resources, Connectors
- Any Model Reference that is a descendant of a BOM Option Class

7.3 Host Application Support of Instantiation

Host applications that are Oracle Applications products must pass the `sbm_flag` parameter in the initialization message to allow multiple instances of BOM Models to be created in a runtime Oracle Configurator. See the *Oracle Configurator Implementation Guide* for information about this parameter and the initialization message.

All host applications support multiple instances of non-BOM Components or non-BOM Model References, regardless of the `sbm_flag` parameter.

In Oracle Applications, after an end user saves a configuration, each BOM Model instance appears as an item under its parent on a separate line in the quote or order.

7.4 Modifying Instantiability

In Configurator Developer, modify a node's Instantiability settings if you want end users to be able to create one or more instances of that component at runtime.

If the Instantiability setting for a Component or Reference node is set to Multiple or Variable Instances, there are two ways to control how many instances are allowed at runtime:

- Modify the node's Initial Minimum and Initial Maximum in Configurator Developer (see Section 29.17.6 on page 29-13)
- Define a Numeric Rule that changes how many instances of the component are allowed at runtime.

  Creating this kind of rule is described in Section 13.7 on page 13-4.

If a node's Instantiability setting is not Multiple or Variable Instances, a Numeric Rule cannot change how many instances of that node can be created at runtime. If such a node is on the right side of the rule, Configurator Developer displays an error when you generate logic.

To edit the Initial Minimum and Initial Maximum values on a nested Reference (that is, a Reference within a Reference), you must open the Referenced Model's parent for editing in the Structure area of the Workbench. For more information, see Chapter 4, "References".

For a list of node types that support multiple instantiation, see Section 7.2.2 on page 7-2.

For information about how changing instantiability settings of node's in a published Model can affect restored configurations, see the *Oracle Configurator Implementation Guide*.

7.5 Loading Models with Instantiable Components

At runtime, an Oracle Configurator window loads the selected Model and the initial number of component instances according to the Instantiability settings defined in Configurator Developer.

Instantiable components that have a large Initial Minimum value may increase the time required to open the Model in an Oracle Configurator window. For details, refer to the *Oracle Configurator Modeling Guide*.

7.6 Runtime Display of Instantiable Components

Configurator Developer provides several UI Content Templates that you can use to display instantiable components at runtime. These templates are described in Section 20.3.3.3, "Instance Management Control Templates" on page 20-17.

For information about how Oracle Configurator creates the default UI captions for instantiable components, see Section 28.9, "Runtime Display Names" on page 28-6.

For more information, see Section A.6, "Creating Instances at Runtime" on page A-5.

8 Connectivity

This chapter describes how to create configuration models in Configurator Developer that allow components to be connected in a runtime Oracle Configurator.

This chapter includes the following sections:

8.1 Introduction

In some Models, the Oracle Configurator end user not only makes selections from the available components, but must also specify how some or all of the components are *connected* before the configuration is valid and complete. For example, when configuring an automobile engine an end user can select either a 6 or an 8 cylinder block, a carburetor, and either low or high-compression pistons, and several other components must be selected for the entire unit to be assembled correctly. The pieces can be connected to form a single unit (the engine), but multiple configurations are possible. And although customers indicate how they want the unit to be assembled, the end result must be something that the factory can produce. A product such as this requires *connectivity* to be defined between specific Model components and rules must exist to ensure that each connection, and ultimately the final assembly, is valid.

Another example of a Model that requires components to be connected is one that enables an Oracle Configurator end user to define a **network**. A network is a system comprised of interrelated or interconnected elements, such as telephones, computers, or television stations. Each kind of network is unique, but all networks are comprised of individual components that are connected in some way.

In Configurator Developer, you build a Model that allows components to be connected at runtime by creating **Connectors**. Connectors define connectivity from a Model or Component node to any other Model in the CZ schema (in other words, any Model that appears in the Main area of the Repository). A Connector can have only one Model as its target. A Model may have one or multiple Connectors, and you can specify whether a connection is required to create a valid configuration. Creating Connectors is explained in Section 29.9 on page 29-5.

If an instance of the Model chosen as the Connector's target exists at runtime, and creating the connection does not violate any configuration rules you have defined, an end user can connect the components at runtime.

As with any Model, Models that allow connections may have constraints defined in Configurator Developer to ensure that all required connections are made, each connection is valid, and all configuration requirements are met.

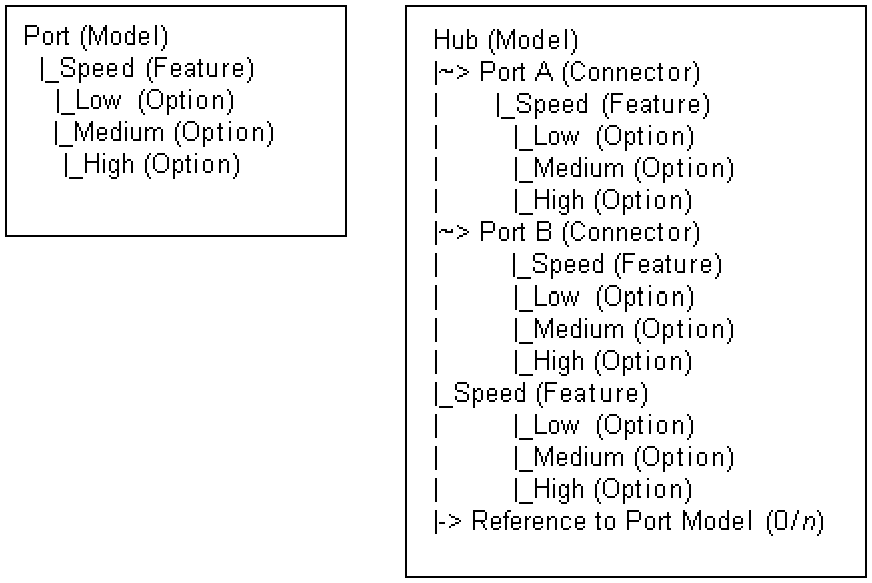
Note: The Generic Configurator UI does not support connecting Model components. An Oracle Configurator end user can connect components only in User Interfaces that you create in Configurator Developer. For more information about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

Example 8–1 Telephone Network Model

In a runtime Oracle Configurator, an end user configures a telephone network by defining circuits. The user creates a circuit by connecting two Ports, which are part of a configurable Hub Model. One or more configurable Ports exist per circuit, and each Port includes the parameter Speed. The Speed parameter includes the Options High, Medium, and Low, and any two connected Ports must have the same speed. The end user connects two Ports at a time, while constraints defined in Configurator Developer ensure that the Speed of the connected Ports is the same. When all required selections and connections are made and the configuration is valid, the data is passed back to the host application for downstream processing.

The Models required to create the network and rules described in this example are shown in Figure 8–1.

Figure 8–1 Telephone Network Models



```
Port (Model)
  |_Speed (Feature)
    |_Low  (Option)
    |_Medium (Option)
     |_High (Option)
```

```
Hub (Model)
|~> Port A (Connector)
|      |_Speed (Feature)
|        |_Low  (Option)
|        |_Medium (Option)
|        |_High (Option)
|~> Port B (Connector)
|        |_Speed (Feature)
|        |_Low  (Option)
|        |_Medium (Option)
|        |_High (Option)
|_Speed (Feature)
|        |_Low  (Option)
|        |_Medium (Option)
|        |_High (Option)
|-> Reference to Port Model  (0/n)
```

Note: Connectivity was designed for users who need to update a large network of connected components, such as a long-distance telephone network. Models that support connectivity, and Models that support reconfiguration of an installed configuration, are implemented primarily by businesses in the Telecommunications Service Ordering industry. For additional information about this functionality, see the *Oracle Telecommunications Service Ordering Process Guide*.

8.2 Connectors and Target Models

In Configurator Developer, you can create a Connector under a BOM Model or a Component node. However, a Connector's **target** is always a Model. In other words, an Oracle Configurator end user can create a connection from:

- A BOM Model to another Model
- A Component to a Model

In Configurator Developer, the Model that is the target of a Connector must be referenced from the Model being configured at runtime. This is because an instance of the target Model must exist in the configuration before an end user can connect the Connector's parent and the target Model (and the only way a Model can be part of another Model's structure when it is a Reference).

Before an Oracle Configurator end user can connect two components at runtime, an instance of each component must exist in the configuration. This is an important consideration when building your Model in Configurator Developer, because all components that you want an end user to be able to connect at runtime must be part of the Model's structure.

If an instance of the target Model does not exist at runtime, the runtime Oracle Configurator displays an empty list of targets when an end user attempts to create a connection. See Section 8.4, "Connectors and the Runtime User Interface" on page 8-7.

8.2.1 Target Model Structure and Rules

When you create a Connector and specify a target Model in Configurator Developer, the read-only structure of the target Model is visible beneath the Connector node and you can use nodes from the target Model when defining configuration rules for the Model in which the Connector is defined. The target Model's read-only rules are also visible when you are working in the Configuration Rules module. To modify the structure of a target Model or any of its rules, you must open the target Model for editing in the Workbench.

Although the target Model and its structure are visible beneath the Connector node in Configurator Developer, a new instance of the target Model is *not* automatically created when another instance of the Connector's parent is added at runtime. If the configuration requires another instance of the target Model, the end user must add it (see Section A.6, "Creating Instances at Runtime" on page A-5). For more information about instantiation, see Chapter 7.

8.3 Connectors and Configuration Rules

A unique characteristic of Models that contain Connectors is that you can define configuration rules relating instances that may be connected at runtime. However, these rules do not propagate until the end user actually connects the components that are participants in the rule. Then, the rule verifies that the connection is valid and propagates to other parts of the Model (depending on the rule's definition). If the connection is valid, the Oracle Configurator end user can continue; otherwise, a contradiction message explains why the connection is invalid. For an example of this behavior, see Section 8.3.2, "Runtime Behavior of Rules Relating Connected Components" on page 8-6.

Before you can create rules to relate components that may be connected at runtime, you must create Connector nodes in Configurator Developer and specify the Model that serves as the Connector's target. (See Section 29.9, "Creating a Connector" on page 29-5.) You can then use any nodes within the structure of the Connector's target Model to define any type of configuration rule.

Connector nodes *themselves* cannot participate in configuration rules, but you can use a Connector node when defining a Configurator Extension Rule. See Section 8.3.3, "Connection Filter Configurator Extension" on page 8-7.

The ability to use nodes within the structure of a Connector's target enables you to create configuration rules between **optional instantiable components**, which are Model or Component nodes that may exist at runtime, but are not required to create a valid configuration (for example, a Component or Model Reference that has an Initial Minimum of 0). For more information, see Section 11.6, "Rules that Relate Components and Models" on page 11-11.

Note: Use caution when using nodes within a target Model as participants in the parent Model's configuration rules. If a node in the target Model is modified or deleted, the rule in the parent Model may become invalid. In this case, Configurator Developer displays an error message when you generate logic for the parent.

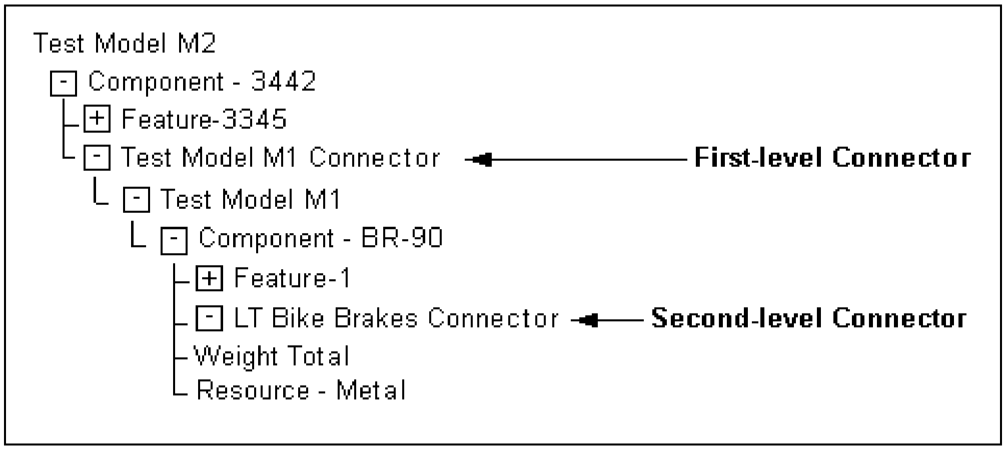
Note: It must be possible for Oracle Configurator end users to connect all necessary components in a configuration in the absence of any previous end user requests or default option selections. For details, and an important Model design suggestion, see the *Oracle Configurator Modeling Guide*.

8.3.1 Second-Level Connectors

A target Model can also contain a Connector. When this occurs the structure of the second-level (nested) Connector's target Model is not visible in Configurator Developer. Because the second-level Connector's target Model is not visible, nodes from that Model cannot participate in the parent Model's configuration rules.

Figure 8–2 shows how a Model containing both first and second-level Connectors appears in Configurator Developer.

Figure 8–2 First and Second-Level Connectors

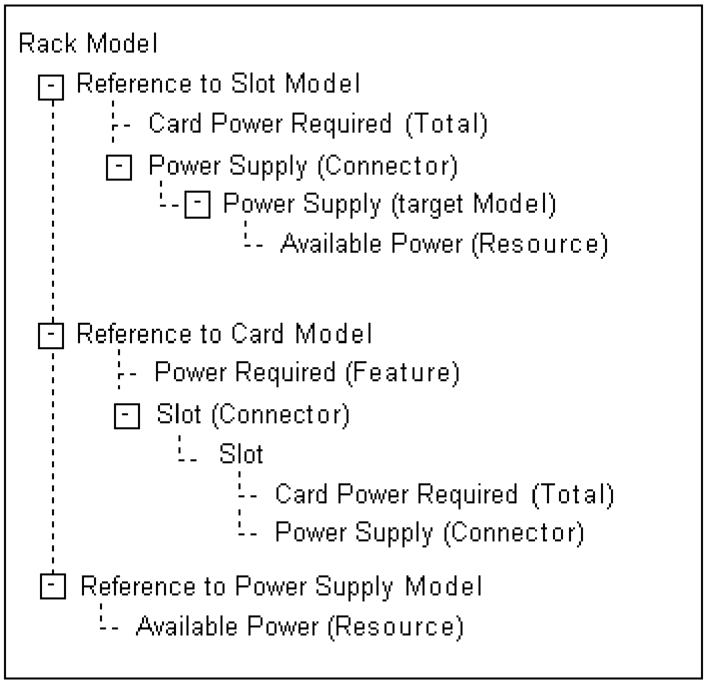


To work around the restriction of creating rules using nodes from a second-level Connector's target Model, perform the following:

1. Create an intermediate node in the first-level target Model. (Feature 1 in Figure 8.2 on page 8-3.)
2. Define a rule in the first-level target that includes the intermediate node and the node(s) you want to use in the nested Connector's target.
3. Define a rule in the root Model using the intermediate node to produce the desired results.

Example: The Rack Model shown in Figure 8–3 contains References to three Models: Slot, Card, and Power Supply.

Figure 8–3 Rack Model Structure


Rack Model
Reference to Slot Model
Card Power Required (Total)
Power Supply (Connector)
Power Supply (target Model)
Available Power (Resource)
Reference to Card Model
Power Required (Feature)
Slot (Connector)
Slot
Card Power Required (Total)
Power Supply (Connector)
Reference to Power Supply Model
Available Power (Resource)

In the Rack Model, Card has a Connector to Slot, which in turn has a Connector to Power Supply. You need to define a rule that states Power Required consumes from the Available Power Resource in Power Supply. In the Rack Model, however, Power Supply is the target of a second-level Connector (in Card), so you cannot use a node from Power Supply to create the rule.

To work around this restriction, define an intermediate node in Slot (in this example, the Card Power Required Total) to reflect information from Card, and then define the rule in two pieces.

First, relate the Power Required Total in Card to the desired node in Power Supply:

```
Card.Power Required Consumes from Power Supply.AvailablePower
```

Then, create a rule in Slot relating Power Required with the intermediate node:

```
Power Required Contributes to Slot.Card Power Required
```

8.3.2 Runtime Behavior of Rules Relating Connected Components

If a rule contains one or more nodes from one or more Connector's target Models, the rule does not propagate until the Connector is connected at runtime.

For example, you want to contribute a constant value to a Total when either Option1 from Component A or Option2 from target Model B is added to the configuration at runtime. You define a "Contributes To" relation (Numeric Rule) and include Option1 and Option2 as participants. This rule will not contribute to the Total, even if Option1 and Option2 are selected, until the end user connects an instance of Component A and an instance of Model B.

8.3.3 Connection Filter Configurator Extension

You may want to control which instances of the target Model appear for selection when an Oracle Configurator user wants to create a connection. To do this, define a **Connection Filter Configurator Extension** and assign it to the Connector node whose target instances you want to filter.

For example, you want to prevent end users from creating more than one connection to an instance of Model X. You can define a Configurator Extension that makes Model X unavailable once the end user creates a connection to it. Then, assign the Connector that has an instance of Model X as its target to the Configurator Extension in Configurator Developer.

To assign a Configurator Extension to a Connector, the Connector must be defined in the Model that is open for editing (that is, not in the Connector's target Model). For example, you are editing Model A, which has a Connector to Model B. Model B also contains a Connector that is visible in the Structure area of the Workbench. In this example, you cannot assign a Configurator Extension to the Connector in Model B when editing Model A.

For more information about creating Configurator Extensions, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

For more information about creating connections at runtime, see Section 8.4 on page 8-7.

8.4 Connectors and the Runtime User Interface

When you generate a User Interface, Configurator Developer uses the UI Master Template you select to create UI controls for each Connector in your Model. For details, see:

- Section 20.3.3.11, "Connection Control Template" on page 20-19
- Section 20.3.3.12, "Connection Chooser Template" on page 20-19
- Section 20.3.3.13, "Connection Navigator Template" on page 20-20

At runtime, when the end user activates a UI control to create a connection, a secondary window displays each available instance of the Connector's target Model. The end user then creates the connection by selecting a specific instance of that Model.

At least one instance of a Model that is the target of a Connector must exist in the configuration before an Oracle Configurator user can connect a component to it.

For example, the minimum number of instances defined for a reference to Model B is 0 (zero). Model B is the target of a Connector that is defined in Model A. When the configuration session begins, the end user cannot connect an instance of Model A to an instance of Model B because no instances of Model B exist. Therefore, until Model B is **instantiated**, no targets are available when the end user clicks the "Create Connection" button in Model A's UI Page.

8.4.1 Connections and Runtime Navigation

When an end user connects two components at runtime, the name of the Model that is the target of the connection appears as a hypertext link next to the "Create Connection" button. Clicking this link navigates to the connected Model's UI Page.

A Connection Navigator control displays a list of components that are connected to a selected component at runtime. Each component appears as a link the end user can click to navigate to that component.

8.4.2 Connecting Hidden Components

A component might not appear in the UI due to a display condition that you define in Configurator Developer, or because the structure node's Display in User Interface setting is not selected in Configurator Developer. In either case, the component is not visible and an end user cannot navigate to it. However, unlike components and options that are not available because of their effectivity, hidden components do exist in the configuration. Therefore, an end user can create a connection, even when the target component does not appear in the UI.

For more information about defining display conditions, see Section 31.3.10 on page 31-27.

Part II

Model Structure

Part II describes each type of node you can create in Configurator Developer and presents information about Populators.

Part II contains the following chapters:

9 Model Structure Node Types

This chapter describes the various kinds of Model structure nodes used when building a configuration model in Configurator Developer.

This chapter includes the following sections:

9.1 Introduction

Populating the CZ schema with data from Oracle Bills of Material creates one or more BOM Models in Configurator Developer. You can open a BOM Model in Configurator Developer and extend it by creating additional structure. You may want to do this if you:

- Want to present guided buying or selling questions to your Oracle Configurator end users
- Need objects to support Totals, Resources, and rules

A Model's structure appears in a table that, when expanded, shows where parent and child relationships exist between product elements. For example, a Component may be a child of a Model, while Options are children of a Feature.

Note: To view a summary of your Model's structure, rules, and Item Master, generate a Model Report. For details, see Section 28.1 on page 28-1.

9.2 Models

Models are described in Section 3.2 on page 3-1.

9.3 BOM Models

BOM Models are described in Section 3.3 on page 3-2.

9.4 BOM Option Classes

A BOM Option Class is typically a group of related BOM Standard Items, but can also contain other, nested BOM Option Classes or BOM Models. BOM Option Classes are imported with a BOM Model from Oracle Bills of Material, and are similar to Configurator Developer Option Feature nodes.

For more information about BOM Option Classes, see the *Oracle Bills of Material User's Guide*. For more information about BOM Models, see Section 3.3 on page 3-2.

9.5 BOM Standard Items

A BOM Standard Item is any Oracle Inventory Item that can be a component on a bill, including purchased items, subassemblies, and finished products. A BOM Standard Item is analogous to an Option in Configurator Developer (that is, a child of an Option Feature).

For more information about BOM Standard Items, see the *Oracle Bills of Material User's Guide*. For more information about BOM Models, see Section 3.3 on page 3-2.

9.6 Components

A Component is a configurable part of a Model and can contain other Components, as well as Features, Resources, Totals, and Connectors. For example, a Computer Model may consist of Components called Hardware, Software, CD-ROM Drive, and Modem. Each of the available choices within these Components are represented by Features (Features are described in Section 9.7 on page 9-2).

To create a Component, see Section 29.2 on page 29-2.

9.7 Features

A Feature can either be selected or accept input (depending on its type) at runtime when configuring a Component. Features can have either a value or enumerated Options. For example, an end user might be able to enter a value between 5 and 20 for an Integer Feature called Length, while an Option Feature called Color might contain Options called Red, Green, Blue, and Yellow.

You can create the following types of Features in Configurator Developer:

- Option Features
- Integer Features
- Decimal Features
- Boolean Features
- Text Features

To create a Feature, see Section 29.3 on page 29-2.

9.7.1 Option Features

This type of Feature contains Options, which appear as children of the Feature node in Configurator Developer. End users select Options at runtime when configuring the Feature. An Option Feature is also known as a List of Options Feature.

An Option Feature's Minimum and Maximum Selections settings indicate how many of the Feature's Options an Oracle Configurator end user can select. For example, if the Feature has five Options and you set the Maximum to 3, only three of the five Options can be selected at runtime.

Additionally, note the following:

- The default value for both the Minimum and Maximum is 1.
- A Minimum of 1 means that at least one Option must be selected. In other words, the Feature is required and must be selected at runtime. If the end user does not select at least one of the Feature's Options, the Feature will be unsatisfied and the configuration will be incomplete.
- A Maximum of 1 means the end user can select only one of the Feature's Options.

  For example, if the end user selects Option 1, and then selects Option 2, Option 1 is automatically deselected.
- A Minimum of 0 means selecting an Option from this Feature is optional.
- You can indicate that a Feature has no maximum value by deleting the numeric value from the Maximum field.

Select the Enable Option Quantities check box if you want end users to be able to specify a quantity for each Option at runtime. For example, if the Options are types of candy bars and the Enable Option Quantities box is selected, the end user can specify six milk chocolate bars and six dark chocolate bars.

If the user must not specify more than one of each Option, do not select Enable Option Quantities.

You can also define a Numeric Rule that uses the number of Options the end user enters at runtime. For details, see Chapter 13, "Numeric Rules".

Note: The Minimum and Maximum Selections settings do not constrain the value an end user can enter when Enable Option Quantities is selected.

You can create Options automatically by defining a Populator or create them manually. For details about Populators, see Chapter 10.

9.7.2 Integer Features

This type of Feature accepts a whole number, such as 3 or 127, at runtime. Integer Features and Decimal Features are often referred to collectively as Numeric Features.

When creating an Integer Feature, you can enter values for the Minimum and Maximum settings. These settings indicate the range of values an end user can enter for the Feature at runtime. The Initial Value is the value you want the Feature to have when the runtime configuration session begins. This value must be between the Minimum and Maximum values, inclusive. For more information about the Initial Value, see Section 9.12 on page 9-6.

A **Count Feature** is an Integer Feature whose Minimum value is greater than or equal to zero. A Count Feature can participate in Numeric Rules (like any other Integer Feature) and Logic Rules, but not Compatibility Rules. An Integer Feature cannot participate in a Logic Rule unless it is a Count Feature.

For information about how Integer Features appear at runtime, see Section 20.3.3.8 on page 20-19.

9.7.3 Decimal Features

This type of Feature accepts a decimal number, such as 3.14159, at runtime. Decimal Features and Integer Features are often referred to collectively as Numeric Features.

When creating a Decimal Feature, the Minimum and Maximum fields indicate the range of values an Oracle Configurator end user can enter for this Feature, in decimal format (for example, 5.0 to 13.5). At runtime, Decimal Features can display up to 9 digits after the decimal point (for example, .123456789).

The Initial Value is the value you want this Feature to have when the runtime configuration session begins. This value must be between the Minimum and Maximum values, inclusive. For more information about initial values, see Section 9.12 on page 9-6.

Decimal Features cannot participate in Logic Rules.

For information about how Decimal Features appear at runtime, see Section 20.3.3.8 on page 20-19.

9.7.4 Boolean Features

A Boolean Feature can have one of the following values at runtime: true (Selected or Auto-Selected), false (Auto-Excluded or Declined), or Unknown (Not Selected). By default, Boolean Features display as an Enhanced Check Box at runtime. See Section 20.3.3.7, "Boolean Feature Check Box Template" on page 20-19.

The Initial Value is the value the Feature has when the configuration session begins. It is also similar to a default value, as it is the value the Feature reverts to when it is not set to a specific value at runtime. For example, a configuration rule states that A implies B. If B has a default value of None, it becomes Logic True when the end user selects A and returns to a logic state of Unknown if A is later deselected. If B has a default value of True, it is initially set to Auto-Selected (Logic True), and retains this state when A is selected and even if A is later deselected.

Additionally, if B is initially True and the end user selects it, it becomes User False (that is, its Detailed Selection State is Declined). For more information, see Section 5.3.1, "Selection State" on page 5-7.

Because Oracle Configurator treats a Boolean Feature's initial value like a default value, it is handled differently than other Feature types when it causes a contradiction. Specifically, if a Boolean Feature's initial value causes a contradiction at runtime, Oracle Configurator ignores it and does not display a contradiction message.

If your UI Master Template uses the Boolean Feature Check Box Control Template to render Boolean Features at runtime, a Boolean Feature with an Initial Value of False is excluded when the configuration session begins (that is, its logic state is Logic False). A Boolean Feature with an Initial Value of True is selected when the configuration session begins (that is, its logic state is Logic True). If your UI displays Boolean Features using a standard HTML check box, the check box is not selected when the Feature's initial value is None or False, and is selected when the initial value is True. (Standard HTML check boxes do not indicate logic state.)

For details about UI Master Templates and displaying logic state at runtime, see Chapter 20, "User Interface Templates".

An Initial Value of None is recommended because choosing either True or False can adversely effect runtime performance.

For more information about initial values, see Section 9.12 on page 9-6.

9.7.5 Text Features

At runtime, this type of Feature appears as an input field that accepts both alphabetical and numeric characters.

The Initial Value you specify in Configurator Developer is the Feature's text when the configuration session begins. For more information about initial values, see Section 9.12 on page 9-6.

You can also control whether an Oracle Configurator end user must enter information for a Text Feature to create a complete and valid configuration. If the Feature is required, an icon appears next to it at runtime. You specify whether a Text Feature is required when viewing the Feature's details page.

9.8 Options

An Option is a child of an Option Feature (see Section 9.7.1, "Option Features" on page 9-3). For example, an Option Feature in your Model called Color includes the Options Red, Blue, Green, and Yellow.

To create an Option, see Section 29.4 on page 29-3.

9.9 Totals and Resources

Totals and Resources can be children of a Model or Component node. A **Total** acts as a numeric variable in your Model (for example, to keep track of a specified quantity), and can have either a positive or negative value. A Total can be used as a constant, or set when an Oracle Configurator end user makes a selection.

A **Resource** is similar to a Total, but a Resource ensures that the quantity it is keeping track of does not fall below zero (that is, become over-consumed). In other words, the runtime Oracle Configurator displays a message when a Resource's value becomes negative, and flags the configuration as invalid until it reaches 0 (zero) or a positive value.

Unlike Numeric Features, Totals and Resources are read-only in a runtime Oracle Configurator, and an end user cannot modify their values. However, you can enable editing of Totals and Resources when unit testing in the Model Debugger. Unit testing using the Model Debugger is described in Section 22.2 on page 22-2.

You can create Numeric Rules that contribute quantities to or consume quantities from both Totals and Resources. For details about using Totals and Resources in rules, see Section 13.7 on page 13-4.

You can enter a decimal value for the Initial Value of a Total or Resource. However, the runtime Oracle Configurator rounds values to two decimal places for these nodes. For example, an Initial Value of 3.12464536 displays as 3.12 at runtime.

To create a Total or a Resource, see Section 29.5 on page 29-3.

9.10 Model References

A Model Reference node (or just Reference) indicates another Model's location in the structure of its parent Model. A Reference can be created manually, in Configurator Developer, or automatically, when you populate the CZ schema with data from Oracle Bills of Material. When you populate the CZ schema with a BOM Model that contains one or more other BOM Models, each child BOM Model within the parent's structure appears as a Reference in Configurator Developer.

For more information about References, see Chapter 4.

9.11 Connectors

In a runtime Oracle Configurator a Connector enables an end user to define connections between component instances. You can create a Connector node under either a Model or a Component node, but the Connector's target must be a Model. A Model can have one or more Connectors.

For general information about Connectivity, see Chapter 8. Creating a Connector is described in Section 29.9 on page 29-5.

9.12 Initial Values

Use this setting to specify a node's value when the configuration session begins (that is, before any quantities are contributed or consumed, and before any rules propagate). By default, the Initial Value for Totals and Resources is blank (null) in Configurator Developer. If you leave this field blank, the initial value displays as a 0 (zero) at runtime.

When the initial value of a Total or Count Feature is zero, configuration rules that involve these nodes do not propagate. Count Features are described in Section 9.7.2 on page 9-3.

For Boolean Features, the initial value is essentially a default, which like all defaults can be overridden by the end user. Therefore, the end user can select a Boolean Feature that is initially Logic False and it will appear as User True in the runtime UI. For more information about Boolean Features, see Section 9.7.4 on page 9-4.

9.12.1 Setting and Updating Initial Values

Oracle Configurator Developer sets initial values when an Oracle Configurator session begins and updates them when, for example, an end user makes a selection.

Oracle Configurator Developer uses the following procedure to set and update initial values:

1. The initial values specified in Configurator Developer are set in the UI.
2. When an end user makes a selection in the Configurator window, the system retracts the initial values. It then changes the selected option to User True.
3. The system applies any rules in which the selection is a participant.
4. The system changes the logic state of any options that are also participants in the rule and are therefore affected by the end user's selection.

There may be situations in which the runtime UI selects an option but the criteria for making the selection is not readily apparent. For example, you define three Boolean Features, F1, F2, and F3 and set the Initial Value for both F2 and F3 to False. You then define a Logic Rule with an Implies relation that states "F1 Implies Any True (F2, F3)."

When the configuration session begins, all of these Features have a logic state of Logic False. When the end user selects F1, the values for all three Features are retracted, become Unknown, and then F1 becomes true. The system then applies the rule and either F2 or F3 becomes false and the other becomes true. This is because the Implies rule states that if the end user selects F1, then either F2 or F3 must also be selected.

If the system selects F2, the user can override it by selecting F3, but there is no setting in Configurator Developer to specify which option should be selected (set to true) and which should be set to false when the rule is initially triggered.

9.13 Numeric Precision and Exponential Notation

To enable configurations to perform precise calculations and support configurations that require a high degree of precision, both Configurator Developer and the runtime Oracle Configurator support both very small and very large numbers.

For example, in your Model:

- The Weight Property of an aluminum washer is .0000005 lbs.
- At runtime, an end user must specify an amount between .0000000001 and 1.0 for a Decimal Feature.
- You need to be able to display very large values using exponential notation (for example, a Total called "Weight" with a value of 10E24 grams)

In Configurator Developer, you can enter constant values using exponential notation for the following:

- The Initial Value of a Total, Resource, or Decimal Feature
- The Minimum and Maximum values of a Total, Resource, or Decimal Feature
- The value of a Decimal Property (that is, the Property’s Data Type is Decimal)
- Values entered when defining a Statement Rule
- Constants in Numeric and Property-based Compatibility Rules

For example, when defining a Decimal Feature you can enter 2.34E-10 as its Initial Value. At runtime, this value appears as 0.000000000234.

10 Using Populators

This chapter describes how to use Populators to create Model structure using data in the CZ schema's Item Master.

Before reading this chapter, you should be familiar with the CZ schema's Item Master. For details, see Chapter 2.

This chapter includes the following sections:

- Types of Nodes Created by Populators
- Moving and Copying Nodes with Populators

10.1 Introduction

You can define a Populator on a non-BOM node to automatically create child structure for that node using Items, Item Types, and Properties in the CZ schema's Item Master. For example, you can create a Populator on Component X and specify the following criteria: "Create Options from Items where the Item is of Type Processor Speed." Running this Populator creates a Feature for each Item that matches the specified criterion. In other words, if there are 10 Items in the CZ schema whose Item Type is Processor Speed, then the Populator creates 10 Features as children of Component X. By default, the nodes that the Populator creates have the same names, descriptions, and Properties as the data used to create them.

When you use a Populator to build Model structure from Items in the CZ schema's Item Master, any Properties and Property values that are associated with the Items are also associated with the new Model structure.

The primary benefit of using Populators is that Configurator Developer maintains a permanent link from the nodes a Populator creates to the source data in the Item Master. Therefore, when data in the Item Master changes, you can update all nodes created using Populators simply by re-running the Populator. This is called "repopulating" the Model. Additionally, if the source data no longer exists in the Item Master, repopulating deletes the corresponding nodes in the Model. Repopulating a Model is described in Section 29.12 on page 29-8.

You can delete a Populator, but it is important to remember that doing so also deletes any Model structure that was created by running the Populator. Deleting a Populator is described in Section 29.13 on page 29-8.

The Properties section on a Model node's details page displays a table with two columns: Inherited and Imported. A check mark in the Inherited column next to a Property indicates that the Property was attached to the node by running a Populator. Inherited Properties can be used in the same way as User Properties that you create manually (for example, when defining rules). For details about imported Properties, see Section 3.3.6, "Item Types and Imported BOM Properties" on page 3-5.

Note: You can repopulate one or more Models without logging into Configurator Developer by running an Oracle Applications concurrent program. For more information, see the *Oracle Configurator Implementation Guide*.

You can also create Model structure using Items and Item Types in the CZ schema's Item Master without using a Populator. Nodes created using this method are also linked to data in the Item Master, but they cannot be easily updated when data in the CZ schema's Item Master changes. Additionally, Properties and their values are *not* incorporated into the Model when you use Item Types to build Model structure. For more information, see Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4.

Creating a Populator is explained in Section 29.10 on page 29-6.

10.2 Types of Nodes Created by Populators

A Populator can create structure on any Configurator Developer node that can have children. These include:

- A non-imported Model
- A Component
- An Option Feature

You cannot define a Populator on a BOM Model, BOM Option Class, or BOM Standard Item. When you run a Populator, it creates new nodes as children of the node on which the Populator is defined.

The type of nodes a Populator can create depends on the node on which it is defined. Table 10–1 summarizes the available choices.

Table 10–1 Populators and Model Nodes

If the selected node is...	You can create...
The root of a non-imported Model	Components, Features (any type), Totals, or Resources
A Component	Components, Features (any type), Totals, or Resources
An Option Feature	Options

10.3 Moving and Copying Nodes with Populators

You can move a node that has one or more Populators from one location in the Model structure to another location in the same Model, or to another Model. When you do this, the node retains its Populator(s). However, when you *copy* a node, Configurator Developer does not assign any of the node's Populators to the new node. In this case, you must manually define the Populator(s) on the new node.

Part III

Configuration Rules

Part III describes the types of configuration rule you can create in Configurator Developer.

Part III contains the following chapters:

11 Rule Basics

This chapter presents general information that you should consider before defining configuration rules.

This chapter includes the following sections:

11.1 Introduction

One of the most critical activities in constructing a configuration model is to design and construct the rules that govern what the end user can select to make a valid configuration. You need to define rules to express relations and compatibilities among the Components, Features, Options, BOM Option Classes, and BOM Standard Items in your Model.

In a configuration model, rules identify Model elements that are:

- Used as general defaults
- Automatically selected when an end user selects another option
- Permitted when an end user selects another option
- Excluded when an end user selects another option

For suggestions about defining rules for best runtime performance, refer to the following documentation:

- *Oracle Configurator Modeling Guide*
- *Oracle Configurator Performance Guide*

11.2 Types of Configuration Rules

Table 11–1 summarizes each type of rule.

Table 11–1 Configuration Rule Types

Rule Type	Description
Logic (see Chapter 12)	Defines a logical relationship between most types of Features, Options, and any type of BOM nodes.
Numeric (see Chapter 13)	Performs a numeric operation on one or more numeric Features, option counts, or Totals placing the result in a numeric Feature, option count, Total, or Resource.
Comparison (see (Chapter 15)	Performs a comparison between the values or Properties of two nodes, or a constant value.
Property-based Compatibility (see Chapter 15)	Specifies matches between the options of one or more Features or BOM Option Classes that have a common Property.
Explicit Compatibility (see Chapter 15)	Specifies matches between the options of one or more Features or BOM Option Classes in explicit tabular form.
Design Chart (see Chapter 14)	Specifies compatibility matches among the options of Features or BOM Option Classes in explicit tabular form.
Statement (see Chapter 16)	Allows more complex expressions and constraint definitions using the Constraint Definition Language (CDL).
Rule Sequences (see Chapter 18)	Specifies an ordered set of rules whose effectivity dates are set so that a rule in the sequence becomes effective at the same time its predecessor ceases to be effective.
Configurator Extensions (see Chapter 17	Use Java code that you write to perform functions that go beyond the functionality and rules that Oracle Configurator Developer provides.

11.2.1 Creating Rules

You create all types of rules (except Java code for a Configurator Extension Rule) in the Rules area of the Workbench. The steps to create a rule vary depending on the rule's type. For details, see Chapter 30.

11.2.2 Rule Folders

In the Rules area of the Workbench, each Model contains a default Configuration Rules Folder. Within this Folder, you can create as many sub-Folders as you need to organize a Model's rules.

Rule Folders that you create can contain any type of rule. You can copy rules and move them from one Folder to another. However, the same rule cannot reside in more than one Folder. Copying a rule to a different Folder creates a new, separate rule that can be modified independently of the original.

To create a rule Folder, see Section 30.12 on page 30-15.

11.2.3 Rule Sequences

A set of rules that are active according to their order in the sequence, which is determined by each rule's effectivity dates. For details, see Chapter 18, "Rule Sequences".

11.2.4 Enabling and Disabling Rules

Enabling and disabling rules can be a useful tool when unit testing and debugging a configuration model. Rules that are disabled are ignored when you generate logic and when you unit test a Model in a runtime Oracle Configurator or the Model Debugger. You can enable or disable any type of rule.

The Disabled column in the Rules area of the Workbench shows whether a rule is currently active. When you enable or disable a rule, Configurator Developer updates this column only after you regenerate logic.

To enable or disable a rule, see Section 30.15 on page 30-16.

11.3 Imported BOM Rules

Basic rules that are inherent in a BOM Model are imported and automatically applied in Oracle Configurator Developer. These rules include settings that indicate whether components in the BOM are optional or mutually exclusive, as well as any **Quantity Cascade** calculations. For example:

- **Required** rules apply to child nodes that become required in the configuration when their parent node is selected. The name of this setting in Configurator Developer is "Required when Parent is Selected." For example, if this setting is Yes for Option Class X, Oracle Configurator automatically selects it when the end user selects its parent BOM Model.
- **Mutually Exclusive** rules apply to nodes that allow only one of their optional child nodes to be selected at a time. The name of this setting in Configurator Developer is "Optional Children are Mutually Exclusive." For example, if the options within a BOM Option Class are mutually exclusive, an Oracle Configurator end user can select only one of them. This is similar to an Option Feature with a Maximum Selections of 1.

  In Oracle Bills of Material, BOM Models that are children (components) of a BOM can be mutually exclusive. However, when you import such a BOM Model, Configurator Developer sets the Mutually Exclusive setting to No for each child (referenced) BOM Model. This is not an issue for most installations, and you can define rules that require a BOM Model's children to be mutually exclusive, if necessary.

Quantity Cascade calculations are explained in Section 11.3.1 on page 11-3.

In Configurator Developer, you can view how these rules are defined by viewing a BOM node's definition in the Structure area of the Workbench. See Section 29.17.5 on page 29-12.

All of an optional BOM Model's required children have an initial logic state of Unknown. Any required child items under the root BOM Model have an initial logic state of true, due to the inherent BOM rules described in the preceding paragraphs. Note that the root BOM Model cannot be optional, and any BOM item that is required has the "Required when parent is selected" setting set to Yes. This setting is described in Section 29.17.5 on page 29-12.

For more information about initial logic states, see Section 11.5.2 on page 11-7.

11.3.1 Quantity Cascade Calculations

Quantity Cascade calculations determine the final quantity requirements for a selected BOM item's children.

When you import a BOM Model, all of the parent-to-child relationships that exist among the components in the BOM are maintained. A parent component (such as a BOM Option Class) may have multiple children; some required, some optional, and some mutually exclusive. When a parent is selected in a runtime Oracle Configurator, all of its required children are also selected. Similarly, when any child is selected, its parent is also selected.

Each component in a BOM is imported with a Minimum Quantity, Maximum Quantity, and Default Quantity. The Minimum Quantity is the smallest number of the selected node allowed per parent. The Maximum Quantity is the largest number of the selected node allowed per parent. The Default Quantity is how many of the selected node will be ordered (per parent) if this value is not modified in the selection process.

Whenever the number of a selected option is greater than zero, a Quantity Cascade calculation is performed which results in the actual quantity (or count) for that BOM item. The Quantity Cascade calculation is:

```
child node actual quantity = (parent node actual quantity) X (child node default
quantity)
```

This calculation is true when the end user selects the parent item (for example, a BOM Option Class) and one of its children (a BOM Standard Item), but does not change the amount of the child item. Therefore, the count of the child is derived from the equation shown above.

However, if the end user enters a different amount for the child (BOM Standard Item) and then changes the amount of the parent (BOM Option Class), the Quantity Cascade calculation is:

```
( new child node amount / current child node amount ) X (old parent node amount) =
new parent node amount
```

For example, Option Class A (the parent node) contains Option1 (the child node). Option1 has an initial count of 2. The end user sets Option1 to 4 and then changes Option Class A to 3. The Quantity Cascade calculation determines a new amount for Option1 as follows:

```
( 4 / 2 ) X 3 = 6
```

Note that the calculation ensures that the new amount of the child node (6) is a multiple of its default quantity (2).

These Quantity Cascade relationships reflect the relationships between components that are built into the BOM to ensure that it is complete and valid before ordering.

Example 11–1 Quantity Cascade Calculation

A BOM Model for an automobile specifies four wheels and five lug nuts for each wheel. If you select the car, that means you must have four wheels and 20 lug nuts. Similarly, if you select one wheel, that forces selection of the car, which forces the Quantity Cascade calculation; therefore, four wheels and a total of 20 lug nuts are ordered. The Numeric Rules you define in Configurator Developer respect these Quantity Cascade relationships.

All quantities for BOM Standard Items are calculated this way and are propagated from the root node down through the entire BOM Model. For more information, see Section 13.5, "Contributing to BOM Item Quantities" on page 13-3.

Initialization Behavior

There are some special considerations for handling BOM quantity values. See the *Oracle Configurator Implementation Guide* for a description of the `model_quantity` initialization parameter.

11.4 Using Node Properties when Defining Configuration Rules

When defining a rule, you select Model nodes that will participate in the rule. A participating node's User Property or System Property can also be part of a rule's definition. For example, you may want selection of an option to decrease the value of Resource X, or increase how many instances of Component A are allowed in the configuration.

When defining a Logic, Numeric, or Comparison Rule, after you select the Model node(s) that participate in the rule, click Choose Properties instead of Apply. Then, select a Property from the list for each rule participant.

User and System Properties are explained in Chapter 5.

For more information about using Properties when defining rules, see:

- Section 13.4, "Using the Model Quantity in Numeric Rules" on page 13-2
- Section 13.5, "Contributing to BOM Item Quantities" on page 13-3
- Section 13.7, "Using Properties when Defining a Numeric Rule" on page 13-4
- Appendix C, "Rules, Node Types, and System Properties" on page C-1

11.4.1 Using Multiple Node Properties in a Rule

There may be times when you need to define a Logic, Numeric, or Comparison Rule, and use more than one of a node's Properties. To do this, you must convert the rule to, or define it from scratch as, a Statement Rule. For example:

Each option within Option Class Y has a User Property called `Memory Supplied`. When one of these options is selected at runtime, you want to contribute the value of the `Memory Supplied` User Property to a Total. You cannot do this simply by defining a Numeric Rule like the one described previously and specifying the `Memory Supplied` Property. This is because the rule's definition requires both the `Selection` System Property and the `Memory Supplied` User Property to contribute the value of `Memory Supplied` to the Total.

To create this expression, you must define the following Statement Rule:

```
Contribute 'Option Class Y'.Selection().Property("Memory Supplied") TO 'Total
Memory Supplied'.Value()
```

For more information about Statement Rules, see Chapter 16, "Statement Rules".

11.5 Configuration Rules and Logic State

To enforce the rules you define in Configurator Developer, the runtime Oracle Configurator maintains a logic state for all selectable options. A selectable option means any part of a Model that can be added to a configuration when configuring a product, either by an end user explicitly selecting something or entering a value in an input field, or by the propagation of a rule. Selectable options in a configuration model include Features, Options, or any optional BOM Models, BOM Option Classes, or BOM Standard Items.

In general, an option's logic state can be:

- True: The option is included in the configuration.
- False: The option is not included in the configuration.

- Unknown: This means that no decision has been made about the option, and it is neither included in nor excluded from the configuration.

  For example, an option is not selected when the configuration session begins or an Integer Feature whose initial value is 0 (zero).

Oracle Configurator and Configurator Developer also use variations of the True and False logic states to identify *how* an option is included or excluded from a configuration. This is because an option can be added to a configuration or excluded from a configuration either by explicit end-user action (for example, selecting an option), or as a consequence of a configuration rule.

When an end user selects an option, its logic state becomes **User True**. When the propagation of a rule causes an option to be selected, its logic state becomes **Logic True**. For example, a Logic Rule states "Option A Implies Option B." When the end user selects Option A, Oracle Configurator selects Option B. Therefore, Option A's logic state is User True while Option B's logic state is Logic True.

An end user or a rule can also exclude an option from the configuration. When an end user deselects an option that was selected by a rule, the option's logic state is set to **User False**. (Deselecting an option that the end user previously selected sets the option's logic state to Unknown.) When an option is excluded by the action of one or more configuration rules, its logic state is set to **Logic False**.

An option's logic state is related to its selection state at runtime. The runtime Oracle Configurator uses this information to determine how to display options and status indicator images in a generated User Interface. For details, see:

- Section 5.3.1, "Selection State" on page 5-7
- Section 20.2.2.7, "Images Section" on page 20-9

Things to Consider

Note that there may be times when an option has a logic state of Logic False but it is still available for selection. This can occur, for example, because of a Defaults Logic Rule or a Maximum Selections relation where the maximum is greater than one. In this case, an option may have a logic state of Logic False but it appears as though it is Unknown (selectable). This is so end users can select it without raising a contradiction.

A BOM Option Class or Option Feature can be either satisfied or unsatisfied at runtime. These nodes are unsatisfied when they are selected (logic state is either Logic True or User True) but none of their children are selected. In other words, the BOM Option Class or Option Feature still requires input or contains at least one required selection.

11.5.1 Generating Logic

At runtime, the Model structure, Model logic, and the runtime UI determine what is available for selection and how information is displayed. Before unit testing or publishing a configuration model, generate Model logic to be sure that it is up-to-date.

Generating logic:

- Loads the Model structure and rules data from the database
- Checks for incomplete rules and inconsistent logic
- Converts the structure and rules data into a format that is usable by the runtime Logic Engine

You generate logic in the General area of the Workbench. For more information, see Section 28.7, "Logic Generation Status" on page 28-5.

11.5.2 Initial Logic State

When a configuration session begins, all options have an initial logic state of Unknown, with the following exceptions:

- Features that have an initial value defined may be either Logic True or Logic False.

  See Section 9.12, "Initial Values" on page 9-6.
- Logic Rules that use the Defaults relation can cause options to be selected (Logic True) or excluded (Logic False).

  See Section 12.1.5, "Defaults" on page 12-4.
- Option Features that have a Minimum Selections set to 1 or greater have an initial logic state of Logic True. This is because at least one of the Feature's Options must be selected to create a valid configuration (in other words, it is required).
- Features that have an initial value defined in Configurator Developer have an initial logic state of Logic True.

11.5.3 Indicating Logic State in the Runtime User Interface

At runtime, Oracle Configurator keeps track of each option's logic state and determines what is displayed to the end user using the following:

- The option's selection state.

  For details, see Section 5.3.1, "Selection State" on page 5-7.
- The images that your UI Master Template uses to indicate selection state and display Enhanced Check Boxes and Enhanced Radio Buttons and at runtime.

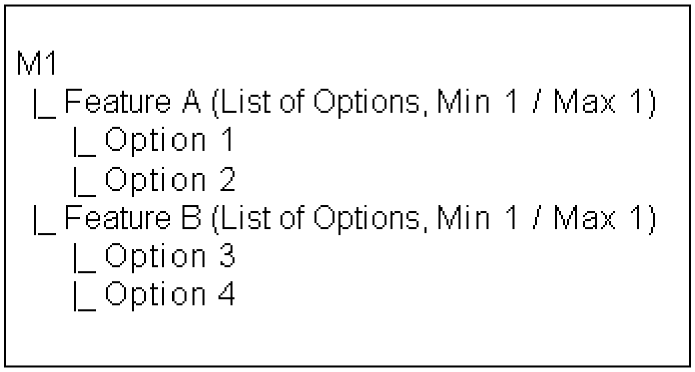
  For details, see Section 20.2.2.7, "Images Section" on page 20-9.

11.5.4 Effectivity and Logic State

Because you can enter an effective date or assign Usages to Model structure nodes in Configurator Developer, there may be times when one or more participants in a rule are not available at runtime. When a node is not available at runtime because of its effectivity, it does not appear in the runtime UI, it is considered by the configuration session to have a logic state of Logic False, and any rules in which the node is a participant behave accordingly.

Consider the Model shown in Figure 11–1.

Figure 11–1 Effectivity and Logic State



```
M1
 |_ Feature A (List of Options, Min 1 / Max 1)
     |_ Option 1
     |_ Option 2
 |_ Feature B (List of Options, Min 1 / Max 1)
     |_ Option 3
     |_ Option 4
```

You assign an effective date to Feature A and create the following Logic Rule:

```
Option 1 Requires Option 3
```

At runtime, the date passed to the configuration session does not match the effective date range assigned to Feature A in Configurator Developer. Therefore, Feature A and its Options do not exist in the configuration, and Option 1 and Option 2 are set to Logic False when the configuration session begins. Option 1 is part of the Logic Rule mentioned above, so Option 3 is also set to Logic False. See Figure 12–3, "The Requires Relation" on page 12-3.

Effectivity is discussed in Chapter 6.

11.5.5 Enforcing Logical Relationships

The rules you create in Oracle Configurator Developer shape the choices an end user can make in the runtime Oracle Configurator. If the end user makes a selection that violates a rule, Oracle Configurator displays a message that describes how the selection affects the configuration as a whole. When a rule is violated, part of the message that appears at runtime is derived from the message you define when creating the rule. For details see Section 30.19.2, "Violation Message" on page 30-18.

For example, an Oracle Configurator end user sets a Numeric Feature to a value that exceeds its defined maximum. Oracle Configurator displays a message similar to the one shown in Example 11–2.

Example 11–2 Maximum Exceeded Message

```
The current value of Inner Diameter is 15. This is above its
maximum of 5.
```

In this case, the end user must enter a new value that is within the defined range for the Feature to continue.

However, Numeric Rules are handled differently. If the user selects values that cause a numeric Feature or Option count to exceed its defined maximum or minimum values, the value of a Total is exceeded, or a Resource is overconsumed, Oracle Configurator displays a message similar to the one shown in Example 11–3.

Example 11–3 Invalid Configuration Message

```
Invalid Items: The current value of Inner Diameter is 15. This
is above its maximum of 5.
```

```
Choose OK to proceed or choose Cancel to return to making your
selections.
```

If the end user attempts to save the configuration in this state, or any other state that violates the rules you have established, Oracle Configurator displays a message similar to the one shown in Example 11–4.

Example 11–4 Unsatisfied Configuration Message

```
Unsatisfied Items: Component BR-90, Test Model M1. Your model is
not complete. There are items that need to be selected or
adjusted. If you continue, the incomplete model will be saved.

Choose OK to proceed or choose Cancel to return to making your
selections.
```

The end user can save the invalid configuration by clicking OK.

If the end user attempts to make a change to the configuration that violates one or more other rules, Oracle Configurator displays a contradiction message similar to the one shown in Example 11–5.

Example 11–5 Contradiction Message

```
There is a contradiction selecting Option A:

You cannot select both Option A and Option C.

If you make this change it will have the following consequences:

Option C will be deselected.

Do you want to make this change?
```

The end user must either agree to reject the last selection or indicate that the change should be made. In the latter case, the system changes values of other Features, Options, and so on until the configuration is restored to a valid state. (In this example, "`You cannot select both Option A and Option C`" is the custom violation message defined for the rule that was violated. For details, see Section 30.19.2, "Violation Message" on page 30-18.)

If an end user's selection triggers a rule that causes an option to become Logic False, it appears as System Excluded in the runtime UI (for details, see Section 11.5.3, "Indicating Logic State in the Runtime User Interface" on page 11-7.) If an end user then selects the option, Oracle Configurator displays a contradiction message, and the end user must click OK or Cancel before proceeding.

Option Features with a Maximum Selections of 1 and BOM Option Classes that have mutually exclusive children that are User False have a Selection State of Selectable. An end user can select such options without receiving a contradiction message. Selection State is explained in Section 5.3.1 on page 5-7.

You can also define conditions for displaying options at runtime. For more information, see Section 31.3.10 on page 31-27.

11.5.6 Unknown Values and Rule Propagation

When the end user starts a new configuration in a deployed Oracle Configurator, many elements in the Model structure have a logic state of Unknown until the user makes a selection. In general, Unknown values within the First Operand side of a rule do not cause the rule to propagate.

If a rule contains multiple participant, and the expression specified involves either the minimum or maximum number of component instances or the operators '+' or '-', the rule propagates when the end user makes a selection. In other words, when a participant within the expression is no longer Unknown.

For example, you create the following Statement Rule:

```
CONTRIBUTE (A + B) TO D
```

If both A and B are Unknown, the rule does not propagate. If A, B, or both options are selected, the rule propagates and contributes a value to D.

If the expression in a rule involves the operators '*' or '/', the rule propagates if one rule participant within the expression is not Unknown and has a value of 0, or if all participants within the expression are not Unknown.

Numeric Rules involving Property values do not propagate unless the option with the Property is selected.

When any node that can have a logic state is a participant in a rule and the value of the option is 0 at runtime, its logic state is Unknown and the rule does not propagate until its value changes. For example, you define the following rule:

```
Option1 = 0 NEGATES Feature2
```

This rule does not propagate when Option1 is set to zero, regardless of whether it is set to zero by default (that is, it has a value of 0 when the configuration session begins) or if it is set to zero by the end user.

For a list of all nodes that have a logic state at runtime, see Section 11.5, "Configuration Rules and Logic State" on page 11-5.

11.5.7 Overriding User Selections without Notification

At runtime, there may be times when the runtime Oracle Configurator deselects one of an end user's previous selections but does not display a message to alert the end user of the change.

Oracle Configurator automatically deselects an option in the runtime UI when both of the following are true:

- An end user selects an option that causes the maximum selections of its parent Feature or BOM Option Class to be exceeded
- Any one of the previous selections was made by the end user (that is, its logic state is User True)

However, Oracle Configurator cannot deselect an option that is a child of a Feature or BOM Option Class if the option was selected by the action of a rule (that is, its logic state is Logic True). If none of the previous selections were made by the end user, no override occurs and Oracle Configurator displays a violation message.

For example, an Option Feature called f1 has a minimum of 1 and a maximum of 2. This Feature contains 3 Options: o1, o2, and o3. The maximum of f1 is reached (2 options are selected) either by end user selection or by the action of a rule. The third option's logic state is Unknown and it is therefore available. When the end user selects the third option, Oracle Configurator may override a previous selection, depending on how the selections were made.

Consider these examples:

- If o1 and o2 are User True and o3 is selected, Oracle Configurator deselects the most recent end user selection and selects o3.

- If o1 is User True and o2 is Logic True and the user selects o3, Oracle Configurator deselects o1.
- If o2 is being contributed to from another item, both o1 and o2 are User True, and the user selects o3, Oracle Configurator deselects o1.
- If both o1 and o2 are Logic True and the user selects o3, there is a contradiction and o3 cannot be made true. In this case, Oracle Configurator displays a violation message.

For more information on logic states, see Section 11.5, "Configuration Rules and Logic State" on page 11-5.

11.6 Rules that Relate Components and Models

Before reading this section, you should be familiar with the concepts and terms presented in Chapter 7, "Instantiation".

There are some restrictions you must keep in mind when constructing rules that relate Components and Referenced Models (this section refers to these nodes collectively as "components"). A rule relates two components if one or more nodes from each component's structure are participants in the rule.

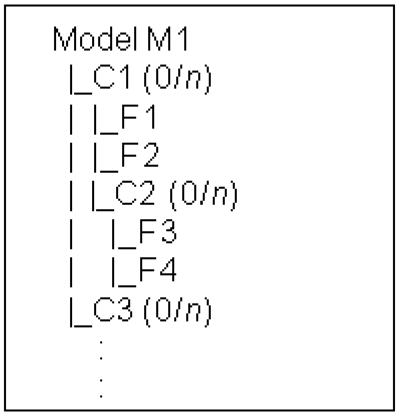
To understand the restrictions that exist when relating optional components, you must understand the following terms that have specific meaning within Oracle Configurator Developer:

- A **required component** is a component whose Instantiability setting is Required Single Instance.
- An **optional instantiable component** is a component whose Instantiability setting is Optional Single Instance.
- An **instantiable component** is a component whose Instantiability setting is Multiple or Variable Instances.

  For details about these settings, see Section 29.17.6, "Instances" on page 29-13.
- The **required substructure** of a component consists of the component itself, all of its required children, all of their required children and so on down to, but not including, any instantiable components.
- Components are **independently instantiable multiple times** when the existence of one instantiable component at runtime does not require the existence of the other. In other words, the components are instantiable and exist at the same level in the Model structure (that is, they do not have a parent-child relationship).

  A rule cannot relate two or more components that are independently instantiable multiple times. In Figure 11–2, C1 and C3 are independently instantiable multiple times, but C1 and C2 are not (because C2 is a child of C1).

Figure 11–2 Component that is Independently Instantiable Multiple Times



```
Model M1
 |_C1 (0/n)
 | |_F1
 | |_F2
 | |_C2 (0/n)
 |   |_F3
 |   |_F4
 |_C3 (0/n)
    .
    .
    .
```

11.6.1 Examples of Valid Rules

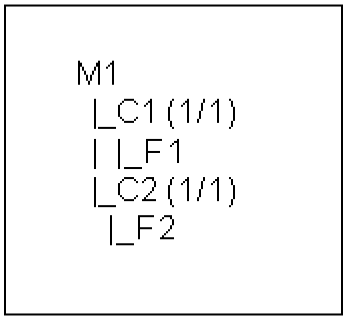
Following are some examples of valid rules that relate runtime components.

11.6.1.1 Rule Relating Components within Required Substructure

A rule is valid if it relates components within a component's required substructure.

Consider the Model structure shown in Figure 11–3.

Figure 11–3 Components within Required Substructure



```
M1
 |_C1 (1/1)
 | |_F1
 |_C2 (1/1)
   |_F2
```

In this Model, "F1 Requires F2" is a valid Logic Rule.

11.6.1.2 Rule Relating Components within Parent's Required Substructure

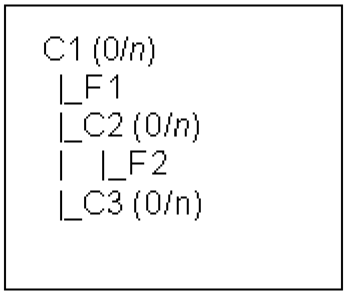
A rule is valid if it relates components within the required substructure of any parent component. In other words, you can create a rule involving any number and any level of components as long as all of them are within the required substructure of a single parent component.

Rule Relating Components that are Instantiable Multiple Times

Consider the structure shown in Figure 11–4. In this Model, C1, C2, and C3 are all Components. C1 is the parent of both C2 and C3. You can create a rule that relates components C1 and C2, or C1 and C3, because any instance of C2 has direct visibility with its only parent (C1) and any instance of C3 has direct visibility with its only parent (C1).

You can create a rule that relates components C2 and C3 only when the Minimum and Maximum instances is 0/1 for both C2 and C3. You cannot create a rule that relates C2 and C3 when their Maximum instances is greater than 1; this is because they exist at the same level in the Model structure.

Figure 11–4 Components within Parent Component's Substructure



In this Model, "F1 Requires F2" is a valid Logic Rule.

Additionally, you can create a valid rule between C2 and C3 if:

- Both C2 and C3 are required (Initial Minimum = 1 or more).
- Both C2 and C3 are optional (Initial Minimum = 0).
- Either C2 or C3 is required, and the other is either optional or instantiable multiple times.

However, you *cannot* create a rule between C2 and C3 if:

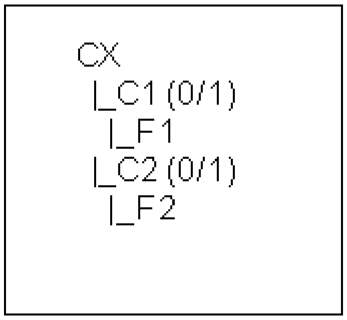
- Both C2 and C3 are instantiable multiple times (for an example, see Section 11.6.2.1 on page 11-14).
- Either C2 or C3 is optional, and the other is instantiable multiple times.

11.6.1.3 Rule Relating an Optional Component with Sibling Optional Components

A rule is valid if it relates an optional component with any number of sibling optional components.

Consider the structure shown in Figure 11–5.

Figure 11–5 Optional Component and Sibling Optional Components



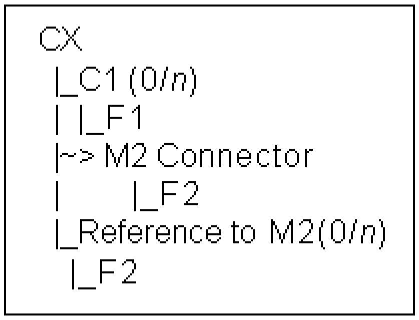
In this Model, "F1 Requires F2" is a valid Logic Rule.

11.6.1.4 Rule Relating Connected Components

A rule is valid if it relates any two components and a Connector connects one component to the other.

Consider the structure shown in Figure 11–6.

Figure 11–6 Component with a Connector



```
CX
 |_C1 (0/n)
 | |_F1
 |~> M2 Connector
 |       |_F2
 |_Reference to M2(0/n)
   |_F2
```

The Logic Rule "F1 Requires F2" is valid if you select F2 from the structure of the Connector's target Model (M2) when defining the rule. The rule is not valid if you select the node from the referenced Model's structure when defining the rule.

For more information about defining rules using structure of a target Model, see Section 8.2, "Connectors and Target Models" on page 8-3.

11.6.2 Examples of Invalid Rules

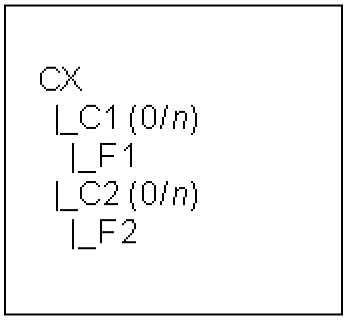
Following are some examples of invalid rule definitions.

11.6.2.1 Rule Relating Sibling Components that are Instantiable Multiple Times

A rule is not valid if it relates two Components that are instantiable multiple times and exist at the same level in the Model structure.

Consider the structure shown in Figure 11–7.

Figure 11–7 Sibling Components that are Instantiable Multiple Times



```
CX
 |_C1 (0/n)
   |_F1
 |_C2 (0/n)
   |_F2
```

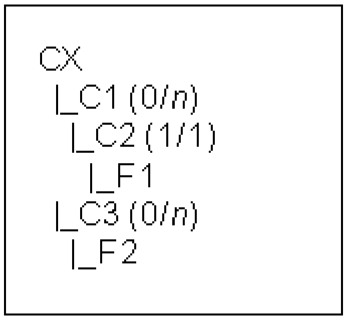
A Logic Rule whose definition is "F1 Requires F2" is invalid.

11.6.2.2 Rule Relating Components within Required, Instantiable Substructure

A rule is invalid if it relates a component that is instantiable multiple times with a component in the required substructure of another component that is also instantiable multiple times and exists at the same level in the Model.

Consider the structure shown in Figure 11–8.

Figure 11–8 Sibling Instantiable Components with Required Substructure



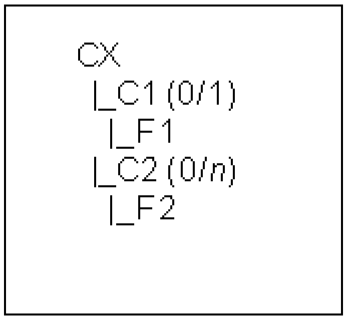
A Logic Rule whose definition is "F1 Requires F2" is invalid.

11.6.2.3 Rule Relating Optional Component with Instantiable Component

A rule is invalid if it relates an optional component with one other component that is instantiable multiple times.

Consider the structure shown in Figure 11–9.

Figure 11–9 Optional Component and a Component that is Instantiable Multiple TImes



A Logic Rule whose definition is "F1 Requires F2" is invalid.

11.7 Unsatisfied Rules

Configuration rules define constraints and requirements for creating a valid configuration and notify the end user when they are violated. When a rule is violated, the configuration is considered invalid, and the end user must make changes before continuing. Configurator Developer also displays a warning message when fewer than the minimum number of options are selected from one or more Features or BOM Option Classes. When this occurs, the configuration is incomplete, and the end user must select additional options to create a complete product.

A third situation can arise when the end user attempts to save a configuration session and one or more options have an Unknown (selectable) logic state, but additional selections are required to create a complete and valid configuration. When this happens, the rule containing the available options is said to be unsatisfied.

Only Logic Rules and Comparison Rules can ever be unsatisfied because they are the only rule types that, at the end of a configuration session, may contain options that are required and have an Unknown logic state.

A Logic or Comparison Rule can become unsatisfied if the rule contains the All True or Any True operators with more than one rule participant. The All True and Any True operators are explained in Section 12.3 on page 12-5.

11.7.1 Examples of Unsatisfied Rules

Example 11–6 Unsatisfied Implies Relation

You define the following Logic Rule:

```
Option A Implies AnyTrue (Option B, Option C)
```

You also enter an unsatisfied message for the rule. At runtime, the end user selects Option A. Option B and Option C still have a status of Unknown, so the rule is unsatisfied. When the end user attempts to save the configuration, the unsatisfied rule message appears. The end user returns to the configuration session and satisfies the rule by selecting either Option B or Option C.

Example 11–7 Unsatisfied Requires Relation

You define the following Logic Rule:

```
AnyTrue (Option B, Option C) Requires Option A
```

You also enter an unsatisfied message for the rule. At runtime, the end user selects Option A. Option B and Option C still have a status of Unknown, so the rule is unsatisfied. When the end user attempts to save the configuration, your unsatisfied rule message appears. The end user returns to the configuration session and satisfies the rule by selecting either Option B or Option C.

Example 11–8 Unsatisfied Negates Relation

You define the following Logic Rule and provide an unsatisfied rule message:

```
AnyTrue ( Option A, Option B ) Negates Option C
```

You create a second Logic Rule with the following definition:

```
 Option Z Negates Option C
```

At runtime, the end user selects Option Z. This gives Option C a status of Logic False, but both Option A and Option B are still Unknown. When the end user attempts to save the configuration, your unsatisfied rule message appears. The end user returns to the configuration session and satisfies the rule by selecting either Option A or Option B.

11.7.2 Unsatisfied Rule Messages

In addition to defining a violation message, you can also define a message that appears when a Logic Rule or Comparison Rule is unsatisfied. Like rule violation messages, an unsatisfied message can contain the rule name, the rule description, or any custom text that you enter.

By default, Logic and Comparison Rules do not display a message when they become unsatisfied at runtime. Therefore, an unsatisfied message is the only way to inform the end user that a rule is unsatisfied and additional options must be selected. In other words, if you do not define an unsatisfied rule message, an end user can save a configuration and will not be informed that additional selections are required.

If you choose to create a custom message, be sure to provide as much information as possible so the end user knows about the available options and how to satisfy the rule by making additional selections.

There may be times when a rule is unsatisfied but it is not necessary to display a message to the end user.

For example, you define the following Logic Rules:

```
All True (A, B) Implies C
```

```
X Excludes C
```

At runtime, the end user selects X, which makes C Logic False. To strictly satisfy Rule 1, at least one of (A, B) would have to be false. In other words, if both A and B are Unknown, Rule 1 is unsatisfied. However, the end user probably does not care whether A, for example, is false or Unknown. Therefore, Rule 1 should have Unsatisfied Message set to None.

In general, you should not specify an unsatisfied message when the rule might be unsatisfied because an option is *Unknown* rather than false.

Creating an unsatisfied rule message is described in Section 30.19.4, "Unsatisfied Message" on page 30-19.

12 Logic Rules

This chapter describes logical relationships and the different types of Logic Rules you can create in Configurator Developer.

This chapter includes the following sections:

12.1 Logical Relationships

Logic Rules enable you to express constraints among elements of your Model in terms of logical relationships. For example, selecting one Option A may require that Options B and C be included in the configuration.

When defining a Logic Rule, you specify the rule’s behavior by selecting a logic relation. Oracle Configurator Developer provides the following logic relations:

- "Implies" on page 12-2
- "Excludes" on page 12-2
- "Requires" on page 12-3
- "Negates" on page 12-3
- "Defaults" on page 12-4

The following sections describe each type of relation and present tables illustrating their behavior. In each table, the unshaded portion indicates the logic state the option has after an end user selects it, and the shaded portion indicates the logic state of the option on the other side of the rule that results from the selection. The arrows indicate the direction in which the rule propagates.

Notice that a rule can propagate from Operand 1 to Operand 2 of the relation, or from Operand 2 to Operand 1. Notice also that for some values and some logic relations the rule does not propagate; therefore logic state of the option on the other side of the rule does not change.

Note: In this chapter, "true" and "false" are used generally to indicate only whether an option is included or excluded from the configuration. Therefore, the examples in the following sections do not differentiate between User True and Logic True, and User False and Logic False. For details about how the runtime Oracle Configurator uses logic state to display options, see Section 5.3.1, "Selection State" on page 5-7.

12.1.1 Implies

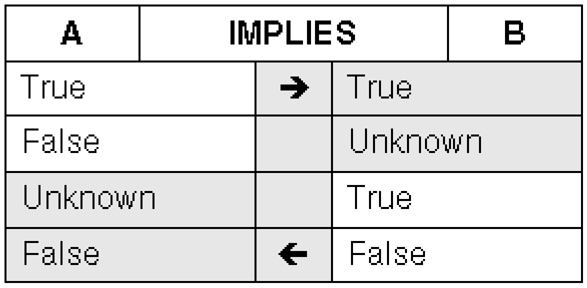
Before reading this section, review Section 12.1, "Logical Relationships" on page 12-1.

The effect of the Implies relation:

- If the end user selects Option A it becomes true and Option B is also selected. In other words, Option B's logic state becomes true.

  See Figure 12–1 on page 12-2.
- Deselecting Option A causes Option B to become Unknown. In other words, Option B is available for selection.
- If the end user selects Option B first, it becomes true and Option A becomes Unknown.
- If the end user deselects Option B, both Option B and A become false.

Figure 12–1 The Implies Relation



A	IMPLIES	B
True	→	True
False		Unknown
Unknown		True
False	←	False

12.1.2 Excludes

Before reading this section, review Section 12.1, "Logical Relationships" on page 12-1.

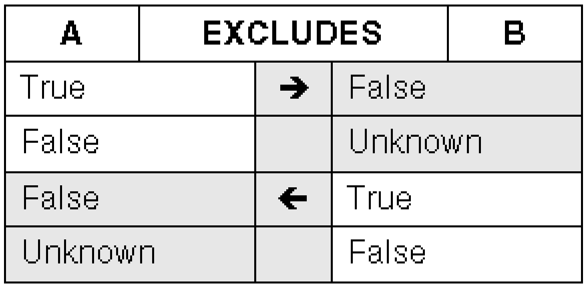
The effect of the Excludes relation:

- If the end user selects Option A, it becomes true and Option B becomes false. In other words, Option B is excluded from the configuration.

  See Figure 12–2 on page 12-3.

  If the end user tries to select Option B, Oracle Configurator displays a contradiction message.
- If the end user deselects Option A, Option B becomes Unknown. In other words, Option B is available for selection.
- If the end user selects Option B first, Option A becomes false.
- If the end user deselects Option B, then Option A becomes Unknown.

Figure 12–2 The Excludes Relation



A	EXCLUDES	B
True	→	False
False		Unknown
False	←	True
Unknown		False

12.1.3 Requires

Before reading this section, review Section 12.1, "Logical Relationships" on page 12-1.

Logic rules that use the Requires relation "push both ways," which means that selecting an option on one side of the rule has the same effect on the option on the other side of the rule. See the examples below for details.

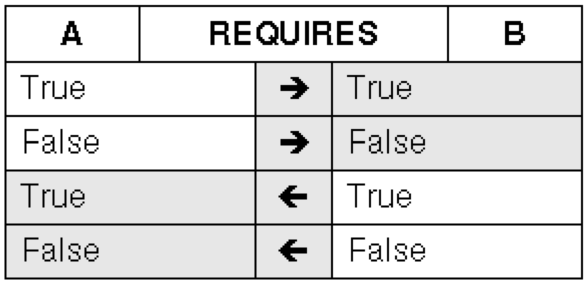
The effect of the Requires relation:

- If the end user selects an option on one side of the rule, the option on the other side of the rule is also selected. The same is true when the end user deselects an option.

  In other words, both options must be either included in the configuration, or excluded from the configuration.

  See Figure 12–3 on page 12-3.

Figure 12–3 The Requires Relation



A	REQUIRES	B
True	→	True
False	→	False
True	←	True
False	←	False

12.1.4 Negates

Before reading this section, review Section 12.1, "Logical Relationships" on page 12-1.

The Negates relation is similar to the Requires relation, in that it also "pushes both ways." However, the Negates relation *prevents* an option from being selected when an option on the other side of the rule is selected. In other words, selecting one option prevents the other option from being included in the configuration.

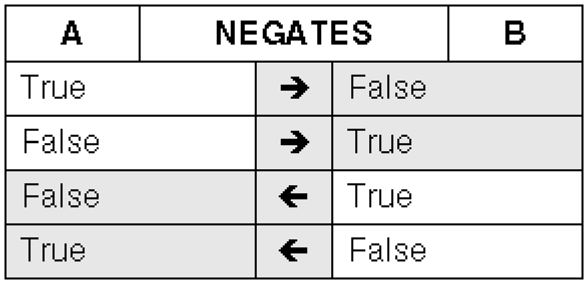
The effect of the Negates relation:

- If the end user selects Option A, it becomes true and Option B is set to false.

  See Figure 12–4 on page 12-4.

- If the end user then deselects Option A, it becomes false and Option B becomes true. In other words, Option B is selected.
- If the end user selects Option B first, it becomes true and Option A becomes false.
- If the end user then deselects Option B, Option A becomes true.

Figure 12–4 The Negates Relation



A	NEGATES	B
True	→	False
False	→	True
False	←	True
True	←	False

12.1.5 Defaults

Before reading this section, review Section 12.1, "Logical Relationships" on page 12-1.

The effect of the Defaults relation: If the end user selects the Option A, it becomes true and Option B is also selected and becomes true.

Unlike other logic relations, the logic state of Option B is not enforced. An end user can set Option B to true or false (that is, select or deselect it), regardless of the state of Option A, and Oracle Configurator will not display a contradiction message. The relation’s only action is to set Option B to true when Option A is true.

If Option B is already true, the rule does not change the option’s state when Option A is selected. In other words, a Logic Rule that uses the Defaults relation only selects *additional* options when the end user, or the action of another rule, selects an option (or options).

A Defaults Logic Rule can be used to set up an initial configuration, for example, by triggering a set of Defaults relations with a Boolean Feature whose initial value is true.

For example, you define a Boolean Feature X with an Initial Value of True. You also define Feature Y which has three Options: A, B, and C. The Maximum Selections for Feature Y is 2. You then define the following Logic Rules:

```
Feature X Defaults Feature Y
```

```
Feature Y Defaults AllTrue (A, B)
```

At runtime, both Option A and B are selected, and have a logic state of Logic True. Because the Maximum Selections for Feature Y has been reached, Option C has a logic state of Logic False. If the end user deselects Option A, it becomes User False and Option B becomes Unknown (that is, it is no longer selected). To avoid this behavior, delete the second rule (Feature Y Defaults AllTrue (A, B)) and define the following rules:

```
Feature Y Defaults A
```

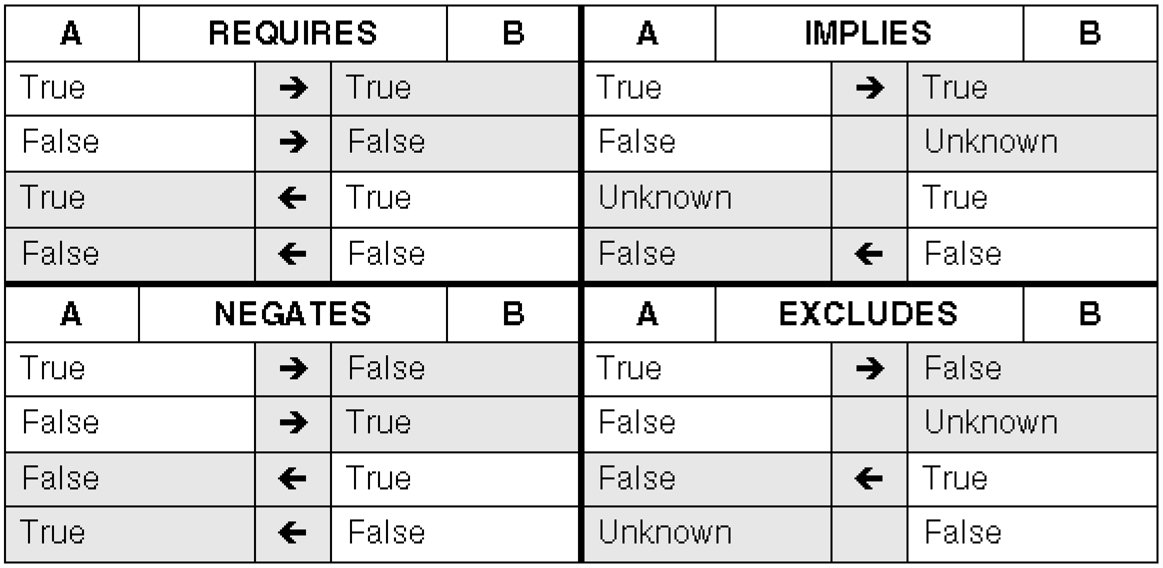
```
Feature Y Defaults B
```

Note: Using many Defaults Logic Rules can significantly affect runtime performance of a configuration model. For more information, see the *Oracle Configurator Modeling Guide*.

12.2 Summary of Logical Relationships

Compare and contrast the effects of the logical relationships as shown in Table 12–5:

Figure 12–5 Summary of Logical Relationships



A	REQUIRES	B	A	IMPLIES	B
True	→	True	True	→	True
False	→	False	False		Unknown
True	←	True	Unknown		True
False	←	False	False	←	False

A	NEGATES	B	A	EXCLUDES	B
True	→	False	True	→	False
False	→	True	False		Unknown
False	←	True	False	←	True
True	←	False	Unknown		False

12.3 Using AllTrue and AnyTrue

The preceding sections describe logical relationships in terms of true and false values for Operand 1 and Operand 2 (the two sides of the relation). You can also define logic relations where Operand 1 or Operand 2 involves an expression of more than one term; for example, all Options of a Feature. In this case, you must specify whether that side of the relation is true if *all* of the terms are true, or if that side is true if *any* of the terms are true.

- Select the AllTrue condition when defining a Logic Rule (or use the AllTrue function when defining a Statement Rule) if you want the relation to evaluate to true only when all terms are true. The relation is false if any term is false. This is a logical AND expression.
- Select the AnyTrue condition when defining a Logic Rule (or use the AnyTrue function when defining a Statement Rule) if you want the relation to evaluate to true if any term is true. The relation is false only when all terms are false. This is a logical OR expression.

  Statement Rules are described in Chapter 16.

13 Numeric Rules

This chapter presents general information about Numeric Rules, such as how they work and ways you can use them when building a configuration model.

This chapter includes the following sections:

13.1 Introduction

Numeric Rules express constraints between elements of your Model in terms of numeric relationships. With Numeric Rules, end-user selections can contribute to or consume from a Resource, Total, Numeric Feature, Option count, or the minimum or maximum number of component instances allowed in a runtime Oracle Configurator.

You can create more complex numeric relationships and constraints by defining a Statement Rule. For details, see Chapter 16.

To create a simple Numeric Rule, see Section 30.4 on page 30-3.

Note: Do not create a Numeric Rule that updates the quantity of the root BOM Model node at runtime. The runtime Oracle Configurator does not support this method of updating the Model quantity (that is, the quantity of the item being configured). The only supported way to update the Model quantity is from the host application. For example, you can change the quantity in the Oracle Order Management Sales Orders window.

13.2 Contributes to Numeric Rules

A **Contributes to** Numeric Rule specifies addition of a value to a specified Numeric Feature, Option count, Total, Resource, or the minimum or maximum number of component instances. Calculation of the numeric value can involve Constants, Boolean values, numeric Features, Option counts, Option Properties, Totals, the minimum and maximum number of instances, and the instance count.

For example:

```
Word Processing, Graphics, Spreadsheet * 60 Contributes to Total 'Disk Space'
```

This rule states that when Word Processing, Graphics, or Spreadsheet is included in the configuration, a value of 60 is added to a Total called Disk Space.

13.3 Consumes from Numeric Rules

A **Consumes from** Numeric Rule specifies subtraction of a numeric value from a specified numeric Feature, Option count, Total, Resource, or instance count. Calculation of the numeric value can involve Constants, Boolean values, numeric Features, Option properties, Totals, the Minimum and Maximum number of instances, and the instance count.

For example:

```
Disk Drive * 10 Consumes from Resource 'Bay Slots Available'
```

This rule states that when the option called Disk Drive is included in the configuration, a value of 10 is subtracted from the Resource called Bay Slots Available.

13.4 Using the Model Quantity in Numeric Rules

To enable the Model's quantity to affect the quantity of other items in a configuration, you must define a Numeric Rule that uses the root Model node's `Quantity` System Property. You must do this, for example, if you want the Model's quantity to be considered during Quantity Cascade calculations for BOM Model items. See Section 11.3.1, "Quantity Cascade Calculations" on page 11-3.

Typically, a user specifies the Model quantity in a host application, such as Oracle Order Management, before launching Oracle Configurator. To unit test this type of rule, launch the Model Debugger and enter a Model Quantity in the Session Parameters page. See Section 32.1, "Unit Testing Using the Model Debugger" on page 32-1.

For example, when an end user orders three computers, you want to ensure that the same number of keyboards are included in the order. To do this, create a Numeric Rule that contributes the Model quantity to the quantity of the Keyboard item.

For example:

```
Sentinal System Model.Quantity * 1 Contributes to Keyboard.Quantity
```

The rule must also include the `Quantity` System Property if your Model contains guided buying or selling nodes and you want to trigger a BOM Quantity Cascade calculation when the non-BOM node is selected. For example:

For example:

```
Option A * 4 * Sentinal System Model.Quantity Contributes to BOM Item S1.Quantity
```

In this case, a value of 4 times the Model quantity is contributed to the imported item S1 when the end user selects Option A. So, if the end user selects Option A and the Model Quantity is 2, the quantity of S1 is 8.

For more information, see Section 13.7, "Using Properties when Defining a Numeric Rule" on page 13-4.

13.5 Contributing to BOM Item Quantities

A common use of Numeric Rules is to contribute to a BOM item’s quantity when an Oracle Configurator end user selects another option, such as a guided buying or selling node. For example, when an end user selects the Custom Laptop BOM item and the "Frequent Traveller" option, an extra rechargeable battery is added to the configuration (in other words, the quantity of the Battery BOM item is set to 2).

To create this type of rule, select a BOM item as a participant on the Second Operand side of the Numeric Rule, and specify the node’s `Quantity` System Property.

For example:

```
Frequent Traveller Option * 2 Contributes to Battery.Quantity
```

All such rules contribute to the BOM item’s total cascaded quantity but not to the unit, or per-parent, quantity. (See Section 11.3.1, "Quantity Cascade Calculations" on page 11-3.) If you have an expression for a contribution to the unit quantity, you must multiply that expression by the parent quantity to get the expected result.

For example, suppose that the Custom Laptop is a component of the Computer System Model. Consider a configuration in which the end user specifies a quantity of 3 for the Custom Laptop and also selects the Frequent Traveller Option. The configuration requires a total cascaded Battery quantity of 6, but the above rule would only contribute 2. To correct this, multiply the First Operand side of the rule by the Custom Laptop quantity. The new rule is expressed as a Statement Rule:

```
CONTRIBUTE (Custom Laptop.Quantity() * Frequent Traveller Option * 2) TO
Battery.Quantity()
```

Because contributions to BOM items are made to the final quantity, it is possible that the contribution will not be divisible by the parent's quantity. In this case, Oracle Configurator finds the closest multiple of the parent's quantity that is less than the contribution. For example, if the parent's quantity is 4 and the contribution toward the child's quantity is 11, the final child's quantity will be 8.

Also, a BOM item's quantity can never be less than its parent BOM item's quantity. Therefore, if the contribution is less than the parent's quantity, then the child's final quantity becomes equal to or greater than the parent's quantity. The exact value depends on the item’s default quantity. See Section 13.5.1, "Default BOM Item Quantity" on page 13-3.

13.5.1 Default BOM Item Quantity

The default BOM item quantity is defined as the minimum BOM component quantity in Oracle Bills of Material. The default quantity is the default quantity of the BOM item when an end user selects it in a runtime Oracle Configurator. Typically, the default quantity for BOM items is 1.

The default BOM quantity is a unit quantity. For example, the default quantity of Standard Item X is 2. An end user sets the quantity of X’s parent to 2, and then selects ABC. This results in a quantity of 4 for Item X.

Contributing to a BOM Item when its default quantity is greater than 1 results in different final quantity depending on the contribution. If the contribution is less than the parent's quantity multiplied by the default BOM quantity, the child's quantity becomes the multiple of the parent's quantity and the child's default quantity.

For example, if a parent's quantity is 4 and the child's default quantity is 1 (which is the same as not specifying a quantity), a contribution of 3 to the child's quantity results in a final quantity of 4 for the child (parent's quantity 4 * child's default quantity 1). However, if in the same example the child's default quantity is 2, a contribution of 3 results in a final quantity of 8 for the child (parent's quantity 4 * child's default quantity 2).

13.6 Using Numeric Features in Numeric Rules

Although it is not common practice, it is possible to define a Numeric Rule in which a Numeric Feature contributes to or consumes from another Numeric Feature. In other words, the participants on *both* sides of the rule are Numeric Features. If you define rules such as this, it is important to understand that the runtime behavior may produce unpredictable results. This is because:

- Oracle Configurator calculates the values for the participant Features in a way that is not apparent to the end user (this is explained below).
- The end user can enter a value for either Feature at any time.

Consider the following rule in which both participants are Numeric Features:

```
Feature A * (Constant 1) Contributes to Feature B
```

At runtime, an end user enters 10 for Feature A, which sets Feature B to 10. Later in the configuration, the end user changes Feature B by entering a value of 20 (this has no effect on Feature A). Then the end user changes the value of Feature A to 30. When this happens, Oracle Configurator sets Feature B to 40.

In the example above, Oracle Configurator initially sets Feature B to 0 (zero). However, when an end user manually enters a value for the Feature on the right side of the rule, the system also maintains an *internal* value for the Feature. This internal value is not visible to the end user and is determined by subtracting the Feature’s old value from the new value (that is, new value - old value). Therefore, when the end user changes the value of Feature B from 10 to 20, this Feature’s *internal* value is set to 10. Then, when the end user changes Feature A to 30, Oracle Configurator contributes 30 to Feature B’s internal value (10) and displays 40 to the end user.

It is generally better practice to use a Total or a Resource on the right side of Numeric Rules, since end users may find the behavior described in the example to be confusing. Totals and Resources are read-only at runtime, so an end user cannot enter a value for the node that (in this example) appears on the right side of the rule.

The behavior described in the example occurs whether the rule uses the Consumes from or the Contributes to relation.

13.7 Using Properties when Defining a Numeric Rule

You can use both System and User Properties when defining Numeric Rules. For an overview of User and System Properties, see Chapter 5. For more information about using System Properties when defining rules, see Appendix C, "Rules, Node Types, and System Properties" on page C-1.

When specifying System Properties on the Second Operand side of the rule, only Properties that can logically be contributed to or consumed from are available. For example, when you add a BOM Option Class node to the right side of the rule, you can select the `Quantity` System Property, but not `State`, `MinQuantity`, or `MaxQuantity`.

Example 13–1 Contributing to the Count of an Option

Each time an Option from Feature X is added to the configuration, you need to ensure that the quantity of Option Y increases by 1. To create this Numeric Rule:

1. Add Feature X to the First Operand side of the rule, and enter 1 as the Quantity Multiplier.
2. Add Option Y to the Second Operand, and click Choose Nodes.
3. In the Choose Nodes page, click **Choose Properties**. Select the `Quantity` System Property from the list, and then save the rule.

The rule now has the following definition:

```
Options{Feature X} * Constant 1, Contributes to Option Y.Quantity
```

Note that in this example, the Enable Option Quantities setting must be selected for the Feature to which Option Y belongs; otherwise, Option Y is simply set to True when an Option from Feature X is selected, since it has no numeric value.

Example 13–2 Contributing to the Maximum Number of Component Instances

When the end user orders a specific quantity of Integer Feature A, you want to increase how many instances of Model Z (a referenced Model) are allowed in the configuration. In Configurator Developer, the Instantiability setting for the Model Z Reference node is set to Multiple or Variable Instances, and the Initial Maximum is set to 3. You define a Numeric Rule in which Feature A contributes to the Model Z Reference node, and specify the Reference node's `MaxInstances` System Property.

The Numeric Rule in this example has the following definition:

```
Total or Numeric Feature{Feature A} * Constant 1,
Contributes to Model Z.MaxInstances
```

At runtime, the end user specifies a quantity of 6 for Feature A. When this occurs, the maximum number of instances of Model Z allowed in the configuration dynamically changes to 9 (that is, a value of 6 is contributed to the Initial Maximum setting, which was 3).

To consume from the *minimum* number of instances allowed at runtime, use the System Property `MinInstances`.

Note: A rule such as the one described in Example 13–2 affects only the minimum and maximum number of instances allowed in a runtime Oracle Configurator. This type of rule cannot actually add instances to the configuration and does not modify the node's instantiability settings in Oracle Configurator Developer.

13.8 Negative Contributions

A negative contribution occurs when the right side of a Numeric Rule is either an Option count or a Count Feature, and:

- The result of the Consumes from relation is a positive value.
- The result of a Contributes to relation is a negative value.

If the sum of contributions is a negative value, Oracle Configurator ignores it. If the sum of contributions is positive, the result is contributed in the ordinary way. For example, if you have a counted option with three rules contributing 5, 4, and -3, the value contributed is 6.

Contributions to a BOM Model are handled similarly. For example, you have a BOM Model with a Quantity count of 3, a BOM Option Class with a Quantity count of 2, and three rules contributing 5, 4, and -3, to the BOM Option Class’s Quantity count. The contribution is calculated as:

```
   5     +     4    +    (-3)    =       6
(rule1)     (rule2)    (rule3)     (contributions)
```

The final count of the BOM Option Class is calculated as:

```
Floor(      6     /      3     )      *     3     =     6
   (contributions) (parent count)     (parent count)
```

The sum of contributions (6) is exactly divisible by the parent count (3), so the final result is 6.

If the sum of contributions is not exactly divisible by the parent count, the result is the closest smaller exact multiple of the parent count. For example, if the contributions are 5, 4, and -4, the result is 3.

If the sum of contributions is smaller than the parent count, then the result becomes the default count. For example if the rule contributions are 5, 4, and -7, the result is 6 (which comes from 2 * 3).

14 Design Charts

This chapter describes Design Charts and how you can use them when building a configuration model in Configurator Developer.

This chapter includes the following sections:

14.1 Introduction

Use Design Charts to express explicit compatibility relationships that are complex and cannot be described using Explicit Compatibility Rules. A Design Chart sets up an array of relationships in the runtime Oracle Configurator so that if the end user selects any of the options included in the Design Chart, that selection can affect the logic state of other options in the Design Chart.

Like other Compatibility Rules, a Design Chart does not select options. Instead, it sets the logic state of any options that are incompatible with the end user's selection to Logic False (which corresponds to the Auto-Excluded selection state), and displays a contradiction message if the end user selects one of the incompatible (excluded) options.

Design Charts provide a simple, efficient way to define the relationship between a Model's *Primary* and *Secondary* Features.

A Primary Feature is an Option Feature or BOM Option Class that defines the variations of the product or service being configured. The compatibilities defined in the Design Chart are based on the Primary Feature. A Secondary Feature can be either a *Defining* or an *Optional* Option Feature or BOM Option Class. A Defining Feature participates in the unique combinations that define the options of the Primary Feature. The options in an Optional Feature can be arbitrarily compatible or incompatible with the options of the Primary Feature. Refer to the example in Section 14.2 on page 14-2 for more information.

Features that are participants in a Design Chart must be Option Features, and they must have the following settings:

- Minimum Selections = 0 or 1
- Maximum Selections = 1

BOM Option Classes that participate in a Design Chart must have the Optional Children are Mutually Exclusive setting set to Yes. For more information, see Section 15.2.1, "Compatibility Rule Participants and Maximum Selections" on page 15-2.

For important information about the default runtime behavior of Design Charts, see Section 15.2.4, "Gated Combinations" on page 15-5.

To build a Design Chart, see Section 30.8 on page 30-8.

14.2 Design Chart Example

In this example, a hypothetical automobile company designs and manufactures several models of sport and full-size pickup trucks. Part of the Model structure appears as shown in Figure 14–1.

Figure 14–1 Example of Automobile Model Structure

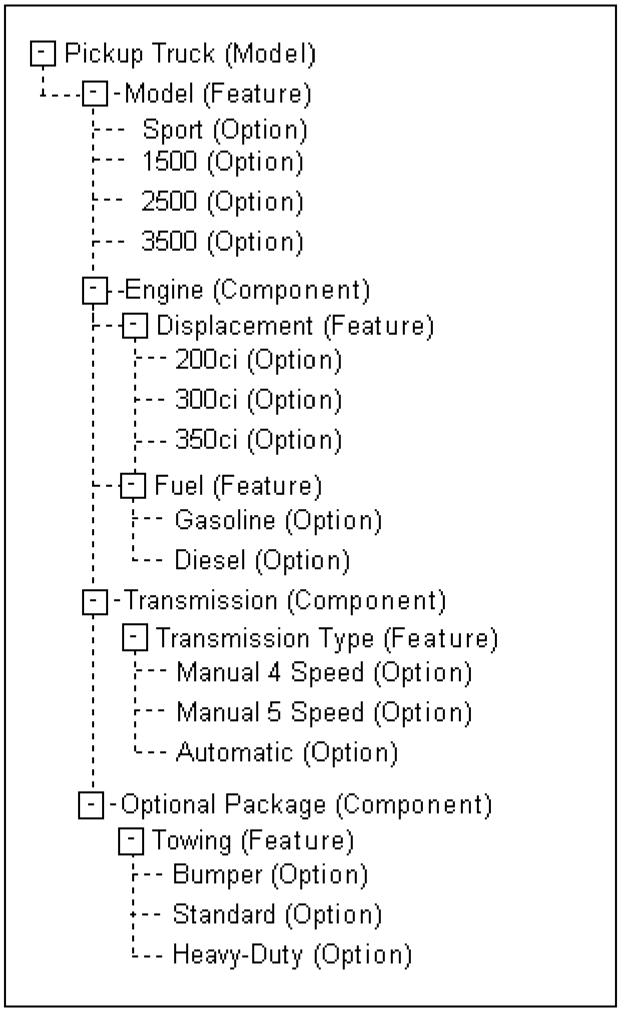


Table 14–1 shows the Design Chart for this product.

Table 14–1 Design Chart for Automobile Model Structure

	Model			
	Sport	**1500**	**2500**	**3500**
Displacement				
200ci	X			
250ci		X	X	
350ci				X
Fuel				
Gasoline	X	X		
Diesel			X	X
Transmission Type				
Manual 4 Speed	X	X		
Manual 5 Speed			X	X
Automatic	X	X	X	X
Towing				
Bumper	X	X		
Standard	X	X	X	X
Heavy-Duty		X	X	X

In this Design Chart, Model is the Primary Feature. Its Options represent the types of trucks that are available. Each truck Model is defined by the engine displacement and fuel used. Therefore, Displacement and Fuel are the Defining Secondary Features. Each of the Primary Feature’s options are defined by a unique combination of Defining Secondary Features.

Only one option of each Defining Feature can be compatible with a given Primary Feature option. When multiple Defining Features are specified, as in this example, the combination of compatible options must be unique for each of the Primary Feature’s options. Each column contains a unique combination of options. Note how the combinations of options specified for Displacement and Fuel are unique for each truck Model.

Various combinations of transmission type and towing package are available for each truck Model, so these are Optional Secondary Features. When the end user chooses the Sport Model, it must have a 200 cubic inch gasoline engine, but it can have either a manual 4 speed or an automatic transmission, and either the Bumper or Standard towing package.

14.2.1 Examples

The tables in this section show three different examples of how the Design Chart shown in Table 14–1 on page 14-3 functions at runtime. Each example assumes the end user has not yet made any selections from the Pickup Truck Model. Note that examples 1 and 2 assume different Minimum Selections values for all participating Features.

Table 14–2 shows the effects at runtime when an end user starts by selecting the Sport Model, and the Minimum Selections of all participating Features is set to 0 (zero).

Table 14–2 Runtime Effects of Selecting Design Chart Options - Example 1

Feature	Option	Logic State
Model	Sport	**User True**
	1500	**Logic False**
	2500	**Logic False**
	3500	**Logic False**
Displacement	200ci	Unknown
	250ci	**Logic False**
	350ci	**Logic False**
Fuel	Diesel	**Logic False**
	Gasoline	Unknown
Transmission Type	Manual 4 Speed	Unknown
	Manual 5 Speed	**Logic False**
	Automatic	Unknown
Towing	Bumper	Unknown
	Standard	Unknown
	Heavy-Duty	**Logic False**

The end user selected the Sport option, so the logic state of this option becomes User True. Because the Maximum Selections for the Model Feature is set to 1, the logic state of the other truck Model options becomes Logic False. Throughout the rest of the Pickup Truck Model, the options that are not defined as compatible by the Design Chart become Logic False. The options that are compatible remain Unknown. (Remember: This example assumes the Minimum Selections on all participating Features is 0.)

The example in Table 14–3 shows the effects at runtime when an end user starts by selecting the Sport Model, and the Minimum Selections for all participating Features is set to 1.

Table 14–3 Runtime Effects of Selecting Design Chart Options - Example 2

Feature	Option	Logic State
Model	Sport	**User True**
	1500	**Logic False**
	2500	**Logic False**
	3500	**Logic False**
Displacement	200ci	Logic True
	250ci	**Logic False**
	350ci	**Logic False**
Fuel	Diesel	**Logic False**
	Gasoline	Logic True
Transmission Type	Manual 4 Speed	Unknown
	Manual 5 Speed	**Logic False**

Table 14–3 (Cont.) Runtime Effects of Selecting Design Chart Options - Example 2

Feature	Option	Logic State
	Automatic	Unknown
Towing	Bumper	Unknown
	Standard	Unknown
	Heavy-Duty	**Logic False**

Note that when the Design Chart constraints leave only one available option, that option becomes Logic True. In this example, these options include the 200ci and Gasoline options.

The example in Table 14–4 shows the effects at runtime when an end user starts by selecting a Defining Secondary Feature, which constrains the Primary Feature. The end user selects a 250cc engine, which limits the available Truck Model choices to 1500 and 2500.

Table 14–4 Runtime Effects of Selecting Design Chart Options - Example 3

Feature	Option	Logic State
Model	Sport	**Logic False**
	1500	Unknown
	2500	Unknown
	3500	**Logic False**
Displacement	200cc	**Logic False**
	250cc	**User True**
	350cc	**Logic False**
Fuel	Diesel	Unknown
	Gasoline	Unknown
Transmission Type	Manual 4 Speed	Unknown
	Manual 5 Speed	Unknown
	Automatic	Unknown
Towing	Bumper	Unknown
	Standard	Unknown
	Heavy-Duty	Unknown

15 Comparison and Compatibility Rules

This chapter describes Comparison and Compatibility Rules, and how you can use them when building a configuration model.

This chapter includes the following sections:

15.1 Comparison Rules

Use Comparison Rules to compare two numeric values to produce a logical result. In this type of rule, two numeric values are compared to determine if the first value is greater than, greater than or equal to, less than, less than or equal to, equal to, or not equal to the second value. The result of this comparison is a logical value (true or false). For example, the numeric value of one Feature or Option compared to the numeric value of another may require that another option be included in the configuration.

To define a Comparison Rule, see Section 30.5 on page 30-4.

For important guidelines to consider when defining Comparison Rules, see the *Oracle Configurator Modeling Guide*.

15.2 Compatibility Rules

Compatibility Rules include Property-based Compatibilities, Explicit Compatibilities, and Design Charts. Each type of Compatibility Rule defines what items are allowed in a configuration, when some other item is selected. A Compatibility Rule compares Options of one or more Option Features, and children of one or more BOM Option Classes (typically BOM Standard Items, but may also include other BOM Option Classes).

Note: In this section, the term "feature" (lowercase "f") refers collectively to Option Features and BOM Option Classes.

Explicit and Property-based Compatibility Rules enumerate all of the allowed combinations of options from the participant features. In other words, if a selection is made from each participant feature and those selections don't correspond to one of the rows of the compatibility table (in an Explicit Compatibility Rule) or satisfy the property-based criterion (in a Property-based Compatibility Rule), there is a contradiction. A Compatibility Rule does not constrain selections from only a subset of the participant features.

Unlike Logic Rules, Compatibility Rules do not actually select anything at runtime. In other words, they do not add an option to the configuration by setting its logic state to Logic True. Compatibility Rules only set the logic state of any options that are incompatible with an end user's selection to Logic False (which corresponds to the Auto-Excluded selection state).

Property-based Compatibilities define compatibility relationships in terms of the values of Properties shared by options within an Option Feature or BOM Option Class. For details, see Section 15.2.2, "Property-based Compatibilities" on page 15-3. Explicit Compatibilities and Design Charts provide two different formats to explicitly define which options are compatible with each other.

For details, see:

- Section 15.2.3, "Explicit Compatibilities" on page 15-4
- Chapter 14, "Design Charts"

For important information about the default runtime behavior of Compatibility Rules, see Section 15.2.4, "Gated Combinations" on page 15-5.

15.2.1 Compatibility Rule Participants and Maximum Selections

A Compatibility Rule's behavior is affected by how many of a participating node's children can be selected at runtime. The action of Compatibility Rules is most clear when each Option Feature involved in the rule has a Maximum Selections of 1, or all of the children within each BOM Option Class are mutually exclusive.

For this reason, Oracle recommends that you use only the following when defining Compatibility Rules and Design Charts:

- Option Features that have a Maximum Selections of 1
- BOM Option Classes that have the Optional Children are Mutually Exclusive setting set to Yes

For example, when an Option Feature involved in a compatibility relationship has a Minimum Selections of 1 in Configurator Developer and all Options but one are excluded at runtime, Oracle Configurator selects the remaining Option (in this case, its selection state is Auto-Selected). Additionally, when an end user selects one of the Feature's Options, the rule excludes the other Options (that is, their selection state is Auto-Excluded). Both examples also apply when an Option Class participates in the rule and all of its optional children are mutually exclusive.

If one or more participant Features have a Maximum Selections greater than 1, or the children of a BOM Option Class are not mutually exclusive, the rule's behavior is more complex and may be confusing at runtime.

For example, Feature 1 and Feature 2 both have a Minimum Selections of 1 and a Maximum Selections of 1. Feature 1 has three Options: Option A1, Option B1, and Option C1. Feature 2 also has three Options: Option A2, Option B2, and Option C2. You define a Compatibility Rule that includes Feature 1 and Feature 2 as participants, and specify the following compatibilities:

- Option A1 and Option A2

- Option B1 and Option B2
- Option C1 and Option C2

At runtime, when the end user selects Option A1, Options B2 and C2 become Logic False. If the end user selects Option B1, Options A2 and C2 become Logic False. If the end user selects Option C1, Options A2 and B2 become Logic False.

Now consider an example when the same Features both have a Minimum Selections of 1 and a Maximum Selections of 3. You define a Compatibility Rule that includes Feature 1 and Feature 2 as participants, and specify the same compatibilities as in the previous example.

In this case, an Oracle Configurator end user is not limited to only one compatible selection for each option. In fact, because the Maximum Selections for both Features is 3, the rule allows *any* combination of its participants.

For more information about the settings described in this section, see:

- Section 9.7.1, "Option Features" on page 9-3
- Section 11.3, "Imported BOM Rules" on page 11-3

15.2.2 Property-based Compatibilities

You create Property-based Compatibility Rules to define compatible combinations of Feature Options or BOM Standard Items based on their Property values. This type of rule can also define compatible BOM Option Classes that are children of a participant BOM Option Class.

Many types of Model structure nodes can have Properties, but only Option Features and BOM Option Classes can participate in a Property-based Compatibility Rule.

When an Oracle Configurator end user makes a selection at runtime, a Property-based Compatibility Rule excludes from the configuration all options that incompatible with the end user’s selection. In other words, the incompatible options are set to Auto-Excluded. For details about selection states, see Section 5.3.1, "Selection State" on page 5-7.

Property-based Compatibility Rules require much less maintenance than Explicit Compatibility Rules because they are updated automatically when any of the participating nodes’ Properties are modified. If both participants in a Property-based Compatibility Rule are BOM Option Classes, the rule is updated when you reimport (refresh) the BOM Model. If the participants are Option Features, the rule is updated immediately when any Properties of either Feature change or, if the Feature has a Populator, when repopulating the Model adds or deletes child Options.

A Property-based Compatibility rule:

- Enumerates the Option Features and BOM Option Classes that participate in the rule
- Specifies the Properties to be checked for compatible values
- Specifies the type of comparison to be made between the Property values, using standard numeric and string comparison operators, such as equals, greater than, contains, and like.

Define a Property-based Compatibility Rule in Configurator Developer when you need to create a single comparison relationship between two structure nodes. To define more than one comparison among multiple nodes, define a Statement Rule using the COMPATIBLE keyword. For details, see Chapter 16, "Statement Rules".

Note: When defining Property-based Compatibility Rules, Oracle recommends that you do not use User Properties whose data type is Translatable Text. Using Properties with this data type in rules can cause the rule to become invalid when the text (the Property's value) is translated.

15.2.2.1 Prerequisites

Before creating a Property-based Compatibility Rule, be sure that:

- Each Option Feature that participates in the rule contains Options, and each Option has one or more Properties.

  Similarly, each BOM Option Class must have children (that is, BOM Standard Items or other BOM Option Classes), and each child must have one or more Properties.

- The children of the nodes that are participants in the rule share at least one common Property

For example, to define a Property-based Compatibility Rule involving Feature1 and Feature2, all of the Options of Feature1 must have at least one Property in common (although the Property *values* may be different), and all of the Options of Feature2 must also have at least one Property in common. (See Example 15–1 on page 15-4.) You can then select the Property that is shared by each Option of a Feature when defining the rule.

Any Properties that are not common to *all* of the Options or BOM Standard Items that participate in the rule are not available when defining the rule.

Example 15–1 Property-based Compatibility Requirement: Common Properties

```
Feature1:
Option1 -
Property Name = Color, Value = Black
Property Name = Weight, Value 16 lbs.
Option2 -
Property Name = Color, Value = White
Property Name = Weight, Value 21 lbs.

Feature2:
Option1 -
Property Name = Wheel Type, Value = Steel
Property Name = Weight, Value 14 lbs.
Option2 -
Property Name = Wheel Type, Value = Alloy
Property Name = Weight, Value 31 lbs.
```

To create a Property-based Compatibility Rule, see Section 30.6 on page 30-6.

15.2.3 Explicit Compatibilities

Explicit Compatibility Rules express compatibility constraints among options of your Model that cannot be described in terms of a shared Property. An Explicit Compatibility Rule allows you to specify, in tabular form, explicit matches between the options of one or more Option Features or BOM Option Classes.

There can be no blank cells in a compatibility table. Options can be repeated in a column as many times as needed to define the available combinations.

For example, consider an automobile model line with several color choices. Table 15–1 lists which colors (Options) belong to each Feature.

Table 15–1 Compatibility Rule Example

Features	Option	Option	Option
Exterior	Red	White	Black
Interior	Tan	Gray	Black
Trim	Gold	Chrome	Black

Some color combinations are available, others are not. The available color combinations can be expressed in a compatibility table as shown in Table 15–2.

Table 15–2 Compatibility Rule Table

Exterior	Interior	Trim
Red	Tan	Gold
White	Gray	Chrome
Black	Black	Black
Red	Gray	Black
Black	Gray	Gold

Each row in Table 15–2 defines a valid combination of exterior, interior, and trim colors. In this example:

- The Red exterior can have a Tan or Gray interior, but not Black.
- The White exterior can have only a Gray interior and Chrome trim.
- The Black exterior can have a Black interior with Black trim, or Gray interior with Gold trim.
- The Gray interior color goes with each exterior color and each trim color, but only three specific combinations of both.

To create an Explicit Compatibility Rule, see Section 30.7 on page 30-7.

Note: For important information about Compatibility Rule participants, see Section 15.2.1 on page 15-2.

15.2.4 Gated Combinations

Gated Combinations refers to the conditions at runtime that cause Oracle Configurator to propagate false in Explicit Compatibility Rules, Property-based Compatibility Rules, and Design Charts. These conditions are determined by a setting in the CZ_DB_SETTINGS table called `GenerateGatedCombo`.

The default value of this setting is `Yes` and it produces runtime behavior that is considered preferable as it causes Oracle Configurator to exclude *more* incompatible selections than it would otherwise. However, if you recently upgraded your Oracle Configurator Developer installation and your Compatibility Rules or Design Charts are producing different and undesirable results, you can restore the old behavior by setting `GenerateGatedCombo` to `No`. For details, refer to the section about the CZ_DB_SETTINGS table in the *Oracle Configurator Implementation Guide*.

15.2.4.1 Behavior Using Gated Combinations

Compatibility Rules are expected to propagate false along a row of an option that is selected (that is, included in the configuration) when all of the following are true:

- There is only one Feature in this Compatibility Rule that is not selected
- The option belongs to the remaining Feature that is not selected
- The option is unavailable (false)
- The option does not belong to another row of the Compatibility table that is invalid

Or, all of the following are true:

- All of the Features of the Compatibility table have been selected
- The option is unavailable (false)
- The option does not belong to another row of the Compatibility table that is invalid

If you recently upgraded your Configurator Developer installation, you may notice that the behavior of Compatibility Rules has changed in the following circumstances:

- One or more participant Features in the rule are optional. In other words, the Feature's Minimum Selections setting is 0 (zero).

  Setting `GenerateGatedCombo` to `No` may force available selections to be false based on the possibly incorrect assumption that the user will eventually make a selection from an optional Feature. But a Compatibility Rule should be interpreted to constrain only *complete sets of selections* from *all* of the participant Features; if an end user makes selections only from a subset of the participant Features, the Compatibility Rule should not affect their selections. That is why this behavior, which is preferable, is the default as it causes *fewer* Options to be set to false than it would otherwise.

  For example, a Compatibility Rule constrains selections from Features X (Minimum Selections = 1 and Maximum Selections = 1), Y (Minimum Selections = 0, Maximum Selections = 1), and Z (Minimum Selections = 0, Maximum Selections = 1). Unless and until the end user makes a selection from Z, or a rule requires that they do so, the end user should be permitted to select Options X1 and Y1 even if no allowed combination of X, Y, and Z contains X1 and Y1.

- The rule includes one or more participant Features with a Maximum Selections greater than one.

  Setting `GenerateGatedCombo` to `No` prevents assumptions based on excluded Options from such Features until the maximum number of selections is reached. A Compatibility Rule should be interpreted to require that every selected Option must be part of a combination allowed by the rule. Any excluded (false) Option from such a Maximum > 1 Feature could be used to rule out Options from the other participants that are compatible only with the excluded Option, even if the maximum number of selections has not been reached.

  The default behavior (`GenerateGatedCombo` set to `Yes`) permits this assumption and causes *more* Options to be forced false.

Note: For important information related to the following example and rules with participant Features that have a Maximum Selections greater than 1, see Section 15.2.1 on page 15-2.

For example, A Compatibility Rule constrains selections from Features A (Minimum = 1 and Maximum = 1) and B (Minimum = 1 and Maximum = 2). The rule allows the following combinations of selections from A and B: {A1, B1}, {A1, B2} and {A2, B3}. In addition, a Logic Rule states that Option X Excludes B3. When the user selects Option X, B3 becomes Logic False. Because of the Compatibility Rule, Oracle Configurator concludes that A2 must also be Logic False, because no other selection from B is compatible with A2.

- The rule includes one participant Feature with Maximum Number of Selections greater than one.

  For example, a Compatibility Rule constraints selections from Feature A (min=1, max=1) and Feature B (min=1, max=2). The rule allows the following combinations of selections from A and B: {A1, B1}, {A1, B2}, and {A2, B1}. If the user selects B1 and B2, the rule cannot exclude A2 (even though A2 cannot be part of the solution) at this point because there is no true propagation. If the end user then selects A2, Oracle Configurator displays a contradiction message. Otherwise, if the end user then selects A1, the configuration will be complete.

In most Models, the default behavior (GenerateGatedCombo set to `Yes`) provides more desirable behavior from the end user's perspective. It avoids invalid assumptions, and makes more valid ones.

15.2.4.1.1 Rules that Depend on Unknown and False Logic States If you upgraded from a previous release of Configurator Developer, the default behavior will affect the outcome of a configuration session only if your Model contains rules that depend on the distinction between Unknown and Logic False.

There are two types of rules that can do this:

- Logic Rules that use the Negates relation, or Statement Rules with operands that contain the NOT operator. The consequences of these rules are propagated when the Negates or NOT operator's operand is Logic False but *not* when it is Unknown. See Example 15–2 on page 15-7.

  Note: This is not the case for rules that use the NotTrue operator.

- Numeric or Comparison Rules involving Numeric Features or Totals with no initial values where the numeric result distinguishes between a value of zero and an Unknown value. See Example 15–3 on page 15-7.

Example 15–2 Propagation of False

A and B are Options. You define the following rule:

```
(NOT A) Implies B
```

In this case, B is True (selected) when A is false but *not* when A is Unknown.

Since it is difficult (and inadvisable) to force all not selected Options to be false, rules of this type are not recommended.

Example 15–3 Rule that Contains a Total with No Initial Value

T is a Total with no initial value. A and B are options. You define the following rules:

```
(A*1) Contributes To T
(T<1) Implies B
```

As in Example 15–2, B is True (selected) when A is false but *not* when A is Unknown.

Note that rules such as the one shown in this example are likely to produce unexpected results and should therefore be avoided. For more information, refer to the *Oracle Configurator Modeling Guide*.

16 Statement Rules

You define a Statement Rule by entering text rather than building the rule interactively by selecting Model structure nodes and operators. A Statement Rule must be written using the **Constraint Definition Language (CDL)**, a rule modeling language that both Configurator Developer and the runtime Oracle Configurator recognize. You can use Statement Rules to define simple to very complex expressions.

Statement Rules can define a Logic or Comparison relationship, a Numeric contribution or consumption, or a Property-based Compatibility relationship. Explicit Compatibilities and Design Charts cannot be expressed using a Statement Rule.

You can also convert an existing Logic Rule, Numeric Rule, Property-based Compatibility Rule, or Comparison Rule to a Statement Rule, and then extend its definition using CDL. After a rule is converted to a Statement Rule and then saved, it cannot be converted back to its original format.

Statement Rules enable you to:

- Write a rule using multiple operands in a single CDL statement
- Include multiple abstract relations in a single rule
- Define both sides of a rule in a single expression

The basic steps for defining a Statement Rule are described in Section 30.9 on page 30-9.

For details about CDL, see the *Oracle Configurator Constraint Definition Language Guide*.

17 Configurator Extensions

This chapter describes Configurator Extensions and how you can use them when building a configuration model in Configurator Developer.

This chapter includes sections on the following topics:

- Configurator Extension Rules
- Configurator Extension Archives and The Archive Path
- Events
- Argument Binding

For details on the procedures related to the topics in this chapter, see:

- Section 25.3.6, "Creating a Configurator Extension Archive" on page 25-4
- Section 28.10, "Configurator Extension Archive Path" on page 28-7
- Section 30.11, "Creating a Configurator Extension Rule" on page 30-10

For examples of how Configurator Extensions are employed, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

17.1 Introduction

Configurator Extensions extend your runtime Oracle Configurator with custom code through established interfaces.

The term *Configurator Extension* includes the following:

- A Configurator Extension *class,* which is the compiled Java code that implements the desired behavior, and which is contained in a Configurator Extension Archive.
- A Configurator Extension *instance,* which is the event-driven execution of the Java binary class at runtime.
- A Configurator Extension *Rule,* which is the arrangements you make in Oracle Configurator Developer to associate the Configurator Extension class with a Model. This includes bindings between method parameters in the Java class and elements of the Model.

The behavior of a Configurator Extension is defined by the methods of a Java class. The Java class is developed outside Configurator Developer, then inserted into the CZ schema in the form of a Configurator Extension Archive.

To enable your Configurator Extension to work with your configuration model, you must associate it with a node in your Model. You create this association in Oracle Configurator Developer as a type of rule called a Configurator Extension Rule.

Configurator Extensions work in any runtime Oracle Configurator. They are triggered either by events during a configuration session or by user-defined commands attached to controls in the user interface.

The Generic Configurator UI, cannot contain visible buttons for triggering Configurator Extensions. For more information about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

17.2 Configurator Extension Rules

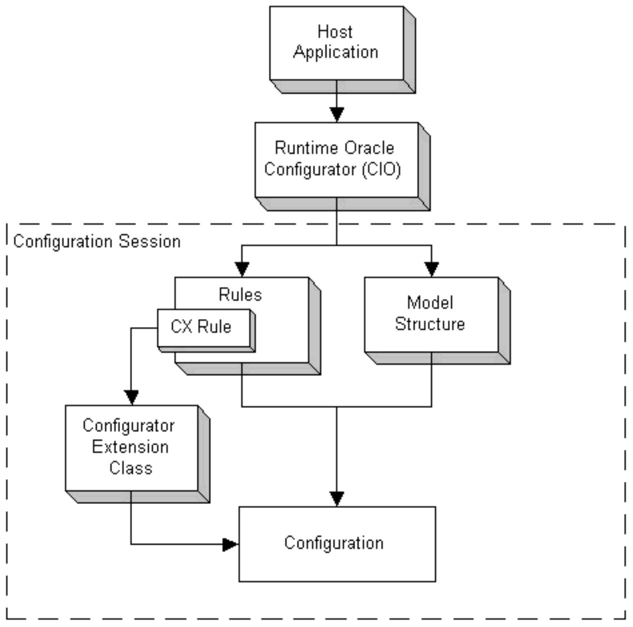
See Section 30.11, "Creating a Configurator Extension Rule" on page 30-10 for the detailed procedure define a Configurator Extension Rule.

A Configurator Extension Rule consists of:

- A specified Model node (called the *base node*).
- An *instantiation scope* for the Configurator Extension (with every base node instance or with the set of instances).
- A specified Java class from a Configurator Extension Archive in the Archive Path for that Model (or from the class path for the host application). See Section 17.3, "Configurator Extension Archives" on page 17-3 and Section 17.3.1, "The Archive Path" on page 17-3.
- One or more *event bindings*, made between a predefined event and a method in the Java class. See Section 17.4, "Events" on page 17-4.
- One or more *argument bindings* for each event binding, between a parameter of the Java method and an argument. See Section 17.5, "Argument Binding" on page 17-9.

During a configuration session, the bound Configurator Extension methods run in reaction to certain configuration session events that are related to specified nodes of the Model. The Configurator Extension can modify the configuration. The relationship of a Configurator Extension to the runtime Oracle Configurator is shown in Figure 17–1 on page 17-2.

Figure 17–1 Relationship of a Configurator Extension to Runtime Oracle Configurator



At runtime, the Java class that implements your Configurator Extension runs in the same JVM as the runtime Oracle Configurator.

17.3 Configurator Extension Archives

See Section 25.3.6, "Creating a Configurator Extension Archive" on page 25-4 for the detailed procedure for defining a Configurator Extension Archive.

A Configurator Extension Archive is an object in the Main area of the Repository that stores one or more compiled Java classes so that they are available both during configuration model development and at runtime. The purpose of an Archive is to incorporate the Java classes that implement Configurator Extensions into the same Oracle Applications environment that is used for developing and running Oracle Configurator. This removes the need to install external class files, configure the Web server to recognize them, and restart the Web server to load them.

17.3.1 The Archive Path

See Section 28.10, "Configurator Extension Archive Path" on page 28-7 for the detailed procedure for defining an Archive Path for a Configurator Extension.

The Archive Path is:

- Like a Java class path for the Configurator Extension Rules defined in a Model
- A setting for a Model that lists the Configurator Extension Archives to be used for Configurator Extensions defined in that Model
- A list whose order determines the order in which the Archives are searched at runtime for the Java classes that implements Configurator Extensions

When you associate a Java class with a Model node (as described in Section 30.11.2, "Choosing the Java Class" on page 30-11), Configurator Developer presents you with a list of all of the classes that are available for use in that association. This list of classes consolidates *all* of the classes in *all* of the Configurator Extension Archives that are in the Archive Path of the current Model.

An Archive can be in the Archive Path of any number of Models. A Model can have any number of Archives in its Archive Path.

17.3.1.1 Archive Path Precedence

The Archive Path determines the precedence among versions of a class. If a class occurs in more than one Configurator Extension Archive, then the list of available classes displays the class stored in the Configurator Extension Archive that is nearest to the beginning of the Archive Path. This resolution of overlapping class names allows you to manage different versions of the same class. To ensure that a certain version of a class is loaded, edit the Archive Path so that the Configurator Extension Archive containing the desired version is ahead of the Archives containing other versions of the class.

At runtime, a parent Model's Archive Path is prepended to every referenced child Model's Archive Path. This means that every Model's Archive Path can be overridden by the parent Model's Archive Path if the parent Model's Archive Path contains a different version of the same class or classes.

At runtime, the Archive Paths of all Models are blended into a single sequence of archives, which is used throughout the entire configuration session. In cases of sibling references to different child Models that contain Archives with different versions of the same class, the class that is defined in the Archive associated with the first child takes precedence, and is used for both Models, even though the second child Model has its own Archive.

In addition to the classes available to a host application through the Archive Paths of its Models, there is almost certainly a set of Java classes that is available through the class path of the host application itself. Since these classes are already in the JVM's class path, they take precedence over the Archive Path.

17.3.2 Using Archives During Development

During development, define your Configurator Extension Archives to reference your Java archive files through a URL so that changes to them are reflected without having to upload them. Later, when your Java classes and your Model are ready to deploy, define your Archive to upload your Java archive files into the database, so that they can be published with the Model.

If you change the signature of a Java method used in a Configurator Extension Rule, then you must create a new binding that reflects those changes. If you have already uploaded the Java class containing the changed method, then you must upload it again.

17.3.3 Using Archives During Deployment

When a configuration model is published, any Configurator Extension Archives that are in its Archive Path are included in the publication, so that the correct version of the uploaded Java class for every Configurator Extension is kept with the Model, and is loaded at runtime when the publication is used to initialize Oracle Configurator. This allows different versions of the same Java class to be running simultaneously in the same JVM. This design also allows you to modify the behavior of a Configurator Extension without restarting the Web server. The ability to change classes without restarting the Web server is sometimes called *hot swapping*.

17.4 Events

An *event* is something that occurs during a runtime configuration session, such a change in the value of a node. Events have names, such as `postValueChange`.

The runtime Oracle Configurator uses the Oracle Configuration Interface Object (CIO) to detect and react to events, using objects called *listeners*, which are registered to listen for the occurrence of specified events. You do not have to explicitly specify listeners when you use Configurator Extensions. When you create an event binding for a Configurator Extension Rule, Oracle Configurator Developer registers the appropriate listener for the specified event.

17.4.1 Event Binding

If an event occurs during a runtime configuration session, and there are bindings for that event with that scope in any Configurator Extension Rules in the Model, then the runtime Oracle Configurator runs all the bound methods for that event.

The events that you can bind to a Configurator Extension are predefined in the CZ schema. Table 17–2, " Predefined Events for Binding" on page 17-6 describes these events. When you define a Configurator Extension Rule, you choose one of these events as part of the binding of your Java class to your Model.

You can define custom events called *command events*. Listeners registered for command events listen for a specified command (defined as a string). You specify the string in a

Configurator Extension Rule as part of an event binding for the event `onCommand`. You can use this functionality to extend the set of events beyond those that are predefined in the CZ schema.

Note: Oracle Configurator does not support any user-defined events other than the custom command event.

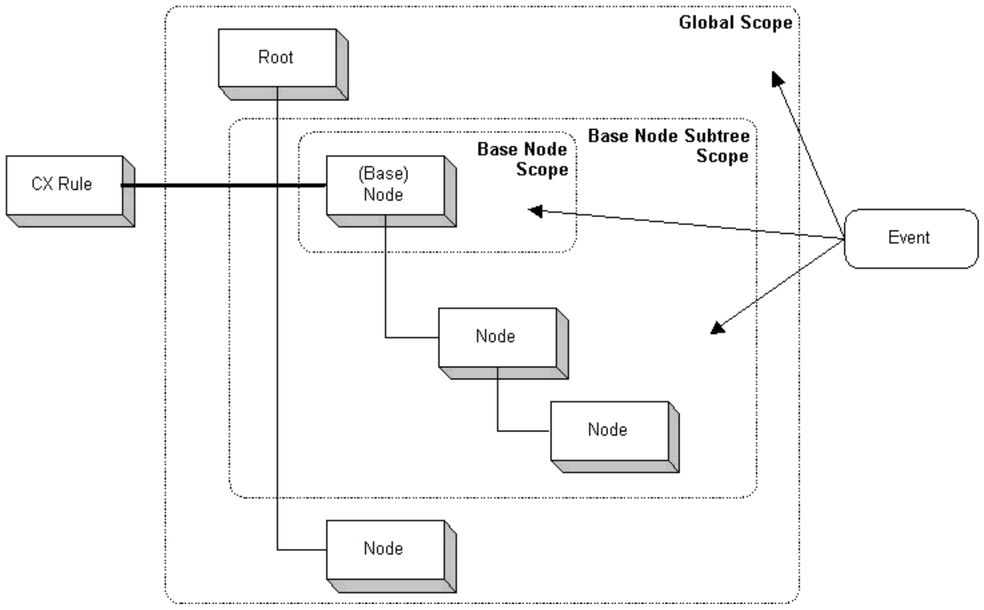
17.4.2 Event Binding Scopes

In Configurator Developer, you bind events within a certain scope. This event binding scope tells listeners that are registered for the event where in the runtime Model tree to listen for an occurrence of that event. Table 17–1 and Figure 17–2, "Event Binding Scopes" on page 17-5 describe the scopes for event binding.

Table 17–1 Event Binding Scopes

Event Binding Scope	Listens for Events ...
Global	Anywhere in the runtime tree of the current configuration instance.
Base Node	Only on the node bound to the Configurator Extension Rule.
Base Node Subtree	On the node bound to the Configurator Extension Rule and all its descendants.

Figure 17–2 Event Binding Scopes



****

At runtime, in a configuration session, an event occurs within an *event execution scope*, which is either Global or Node.

- Events that occur in the Global scope are propagated to all listeners whose event scope is Global. These events are the ones in Table 17–2 on page 17-6 that are related to the configuration object, such as `postConfigInit`.

- Events that occur in the Node scope are propagated in the following sequence, which ascends the tree structure of the configuration model:
    1. By listeners that are registered on the node of execution scope and that have an event binding scope of Base Node or Base Node Subtree
    2. By listeners that are registered on *ancestors* of the node of the execution scope and that have an event binding scope of Base Node Subtree (which means that the node of execution scope is part of their Base Node Subtree binding)
    3. By global listeners, which are registered on all nodes.

17.4.3 Predefined Events for Binding

Table 17–2 on page 17-6 describes the predefined events that you can bind to a Configurator Extension. Some events have event-specific parameters that you use as arguments when binding the method parameters of a Java class in a Configurator Extension Rule. These parameters are included in Table 17–2.

See Section 30.11.3, "Creating Event Bindings" on page 30-12 for the detailed procedure for defining the event bindings for a Configurator Extension Rule.

Table 17–2 Predefined Events for Binding

Event Name	Related To	Description	Event Parameter Name and Type	Event Binding Scope
`postCXInit`	Configurator Extension	Event dispatched after initialization of the Configurator Extension. This can occur on initialization of a configuration session, or runtime addition of a component.	None	Base Node only
`preCXTerminate`	Configurator Extension	Event dispatched immediately before termination of the Configurator Extension. This can occur on termination of a configuration session, or runtime deletion of a component.	None	Base Node only
`postInstanceAdd`	Component Instance	Event dispatched immediately after adding a component instance.	`compSet` (ComponentSet) `instance` (Component)	All scopes
`postInstanceDelete`	Component Instance	Event dispatched immediately after deleting a component instance.	`compSet` (ComponentSet) `instance` (Component)	All scopes
`postInstanceEditable`	Component Instance	Event dispatched immediately after making a component instance editable.	`instance` (Component)	All scopes
`postInstanceNonEditable`	Component Instance	Event dispatched immediately after making a component instance non-editable.	`instance` (Component)	All scopes
`postInstanceNameChange`	Component Instance	Event dispatched immediately after a component's instance name is changed.	`instance` (Component)	All scopes

Table 17–2 (Cont.) Predefined Events for Binding

Event Name	Related To	Description	Event Parameter Name and Type	Event Binding Scope
`onInstanceLoad`	Runtime Node	Event dispatched when a component instance or other associated node is created, or brought into the configuration.	`newNode` (IRuntimeNode)	All scopes
`postInstanceLoad`	Runtime Node	Event dispatched immediately after a component instance or other associated node is created, or brought into the configuration.	`newNode` (IRuntimeNode)	All scopes
`onValidateEligibleTarget`	Connection	Event dispatched during the validation of eligible Connector target instances. A method bound to this event should return a Boolean value. Oracle Configurator uses this value to decide whether the Event Parameter `target` is eligible. If the bound method does not return any Boolean value, then it assumes that `target` is eligible.	`connector` (Connector) `target` (Component)	Base Node only
`postConnect`	Connection	Event dispatched immediately after a Connector is connected to a target component instance.	`connector` (Connector) `target` (Component)	All scopes
`postDisconnect`	Connection	Event dispatched immediately after a Connector is disconnected from a target component instance.	`connector` (Connector) `target` (Component)	All scopes
`onCommand`	Custom Command	Event dispatched when Oracle Configurator runs a custom-defined command. This event must be used when generating custom output.	`HttpServletResponse` (HttpServletResponse)	All scopes
`postConfigInit`	Session	Event dispatched immediately after initializing a new or restored configuration session.	None	Global only
`postConfigNew`	Session	Event dispatched immediately after a new configuration session has been initialized.	None	Global only
`postConfigRestore`	Session	Event dispatched immediately after a restored configuration session has been initialized.	None	Global only

Table 17–2 (Cont.) Predefined Events for Binding

Event Name	Related To	Description	Event Parameter Name and Type	Event Binding Scope
`preConfigTerminate`	Session	Event dispatched immediately before terminating a configuration session. In the UI, this occurs after the end user clicks either the "Done" or "Cancel" button and then clicks an "OK" button to dismiss any notifications or warnings that might be displayed by Oracle Configurator.	None	Global only
`preConfigDone`	Session	Event dispatched immediately before completing a configuration session. In the UI, this occurs after the end user clicks the "done" button and then clicks an "OK" button to dismiss any notifications or warnings that might be displayed by Oracle Configurator.	None	Global only
`preConfigSave`	Session	Event dispatched immediately before saving a configuration.	None	Global only
`postConfigSave`	Session	Event dispatched immediately after saving a configuration.	None	Global only
`preConfigCancel`	Session	Event dispatched immediately before canceling a configuration session. In the UI, this occurs after the end user clicks the "cancel" button and then clicks an "OK" button to dismiss any notifications or warnings that might be displayed by Oracle Configurator.	None	Global only
`preConfigSummary`	Session	Event dispatched immediately before displaying the Summary of a configuration session.	None	Global only
`onConfigLineType`	Session	Event dispatched during the calculation of a line type, which occurs when saving a configuration or displaying its summary.	None	Global only
`onConfigValidate`	Values	Event dispatched during the validation performed after every CIO transaction.	None	Global only
`postValueChange`	Values	Event dispatched immediately after the value of a node is changed.	`changedNode` (IRuntimeNode)	All scopes

Note: You cannot specify an argument whose type is ComponentSet by selecting a node in your Model. You must specify the argument by selecting the event parameter `compSet`. See Table 17–2, " Predefined Events for Binding" on page 17-6.

17.5 Argument Binding

To complete an event binding in a Configurator Extension Rule, you must bind each parameter of the Java method that implements the desired behavior to an argument. The argument specifies some object or value that is available during the configuration session.

Table 17–3 describes the types of arguments that you can bind to method parameters.

Table 17–3 Parameter Types for Argument Specification

Parameter Type	Meaning
System Parameter	An object obtained from Configurator Developer or the runtime Oracle Configurator, containing information about the runtime environment of the Configurator Extension. Table 17–4, " System Parameters for Argument Specification" on page 17-9 describes these objects.
Event Parameter	A parameter of the event that is bound to the method. Event parameters are specific to particular events. Some events have no parameters, if the nature of the event does not require them. Table 17–2, " Predefined Events for Binding" on page 17-6 describes event parameters for the events that are predefined in the CZ schema.
Model Node or Property	A node of the Model to which the Configurator Extension is bound, or a Property of one of the nodes.
Literal	An unquoted character string. The string is automatically converted to an integer or decimal if the method parameter requires that type.

Table 17–4 describes the parameters that you can choose when you select to bind a System Parameter to an argument.

Table 17–4 System Parameters for Argument Specification

Parameter	Meaning
BaseNodeOfRule	The Model node that is bound to the Configurator Extension Rule.
CXEvent	An instance of the `CXEvent` class in the CIO, which contains accessor methods for obtaining information about the event that triggered the Configurator Extension Rule.
CXRule	An instance of the `CXRule` class in the CIO, which contains accessor methods for obtaining details about the Configurator Extension Rule itself.
Configuration	The configuration object created by the CIO during the runtime session in which the Configurator Extension runs. This is an instance of the `Configuration` class in the CIO, which contains accessor methods for obtaining details about the configuration.
Rule Description	The text of the Description of the current Configurator Extension Rule. You create this Description when you create the Rule. You can also obtain this data through the CXRule parameter.
Rule ID	The internal Rule ID of the current Configurator Extension Rule. This ID is generated automatically by Oracle Configurator Developer when you create a Rule. You do not need to know its value when using this System Parameter. You can also obtain this data through the CXRule parameter.
Rule Name	The Name text of the current Configurator Extension Rule. You create this Name when you create the Rule. You can also obtain this data through the CXRule parameter.

See Section 30.11.4, "Binding Arguments to Parameters" on page 30-13 for the detailed procedure for defining the argument bindings for a Configurator Extension Rule.

17.6 Legacy Functional Companions

By default, you cannot create or maintain Functional Companions in Configurator Developer, as they have been replaced by Configurator Extensions. However, if you are still using DHTML UIs from a previous release and need to create or maintain Functional Companions, your system administrator can provide this functionality by changing the value of a profile option.

For more information, refer to the Upgrade Considerations section in the *About Oracle Configurator* documentation, which is available on Metalink.

18

Rule Sequences

This chapter describes Rule Sequences and how you can use them when building a configuration model in Configurator Developer.

This chapter includes the following sections:

18.1 Introduction

A Rule Sequence is a set of rules that are ordered sequentially according to their effective dates. Any type of rule can participate in a Rule Sequence, including Configurator Extensions.

A rule in a Rule Sequence becomes active only when its predecessor becomes inactive. Inactive rules are ignored at runtime. Create a Rule Sequence when you want to automatically deactivate one rule and activate another at a specific point in time. Effective dates are explained in Chapter 6, "Effectivity".

You define effective dates for each rule in a Rule Sequence to determine when each rule in the set becomes active, how long it is used, and when it becomes inactive. Since a rule must become active as soon as its predecessor becomes inactive, modifying the effective date of one rule in a Rule Sequence can affect the effective dates of at least one other rule in the set. Each rule in a Rule Sequence can either have an effective date range defined or be assigned to an Effectivity Set. See Section 18.4, "Rule Sequences and Effectivity Sets" on page 18-2.

When you create a new Rule Sequence, it contains no rules. You must then add rules to the Rule Sequence either by moving existing rules into the Rule Sequence, or by creating new rules within the Rule Sequence. By default, the effectivity of the first rule you add to a Rule Sequence is Never Effective. You can then modify the rule's start and end dates and add other rules to the sequence. Each rule that you add to a Rule Sequence appears at the end of the sequence and is also Never Effective by default.

The effective dates of rules in a Rule Sequence cannot overlap. When you modify the effectivity of a new rule in a Rule Sequence, Configurator Developer ensures that the new rule's effectivity does not overlap with any other rules in the sequence.

You can delete a Rule Sequence, but doing so also deletes all of the rules it contains from the configuration model.

Creating Rule Sequences is explained in Section 30.10 on page 30-9. Removing rules from a Rule Sequence is described in Section 30.17 on page 30-17.

18.2 Viewing Rule Sequences

You can view and modify Rule Sequences in the Rules area of the Workbench. Configurator Developer uses a unique icon to distinguish Rule Sequences from rules. Rules that belong to the Rule Sequence appear as children of the Rule Sequence node, and are listed in the order in which they appear in the sequence. This order corresponds to the chronological order in which each rule in the set becomes active over time.

A Rule Sequence's details page lists all rules that the Rule Sequence contains. From this page you can reorder rules in the sequence, or edit a rule's details (such as its effective dates). For details, see Section 30.16, "Reordering Rules in a Rule Sequence" on page 30-16.

18.3 Modifying the Effectivity of a Rule in a Rule Sequence

After adding a rule to a Rule Sequence, you can change the rule's effective dates or modify the Effectivity Set to which it is assigned. When you do this, Configurator Developer adjusts the date ranges of the rules that precede and follow that rule (if any) to maintain the constraints inherent in the sequence. If this is not possible, Configurator Developer displays an error and does not change the rule's effectivity.

Following are a few examples of what occurs when you modify effective dates for rules in a Rule Sequence:

- You assign a start date to the first inactive rule in the sequence, or modify the start date of an active rule. Configurator Developer adjusts the end date of the preceding rule (if any) to match. If it cannot do this without changing the dates of other rules in the sequence, Configurator Developer displays an error message and rejects the new date.
- You modify the end date of an active rule, and the following rule is active, Configurator Developer adjusts the start date of the rule that follows. If it cannot do this without changing the dates of other rules in the sequence, Configurator Developer displays an error message and rejects the new date.
- You deactivate the last active rule in a sequence by setting it to Never Effective, and that rule has a predecessor. Configurator Developer sets the end date of the predecessor to what was previously the deactivated rule's end date.
- You deactivate a rule that is not the last active rule in the sequence. Configurator Developer displays an error message. To deactivate a rule, you must move it after the last active rule in the sequence, and then set it to Never Effective.
- You assign start or end dates to an inactive rule whose predecessor is not active. Configurator Developer displays an error message because the first rule that is active must appear first in the Rule Sequence.

18.4 Rule Sequences and Effectivity Sets

To learn about Effectivity Sets, see Section 6.3 on page 6-2.

When you modify an Effectivity Set's start and end dates, and the Effectivity Set contains members of one or more Rule Sequences, Configurator Developer displays any error conditions associated with Rule Sequence date constraints and rejects the new date(s). If there are no errors, Configurator Developer applies the new dates to each rule and adjusts the effectivity dates of each rule in the Rule Sequence as necessary.

Sometimes modifying a Rule Sequence requires a change to a rule’s effective dates that conflicts with the effective dates of the Effectivity Set associated with the rule. In this case, the rule whose effective dates need to be changed will be disassociated from its Effectivity Set and the new effectivity dates will be applied directly to the rule itself. When this occurs, Configurator Developer displays a warning message and you can choose to either accept the new effectivity dates, or cancel the operation.

Some examples of how this situation can occur include:

- Activating the first inactive rule in a Rule Sequence when its predecessor is a member of an Effectivity Set.
- Adding a rule to a Rule Sequence in a position such that its new predecessor and/or successor is active and is a member of an Effectivity Set.
- Removing a rule from a Rule Sequence if its predecessor and successor is active and a member of an Effectivity Set.
- Moving a rule to a location within a Rule Sequence in which either of the previous two conditions apply.
- Adding a rule to a Rule Sequence or moving a rule within a Rule Sequence, if the rule being added or moved is itself a member of an Effectivity Set and its new position in the Rule Sequence requires a change to its own effective dates.

18.5 Reordering Rules and Rule Effective Dates

Following are some examples of how effective dates may change when you reorder rules in a Rule Sequence:

- If the rule that you move is inactive (that is, never effective and appearing at the end of the sequence), its start and end date are adjusted to be consistent with its new position.
  - If its new position is at the end of the sequence or its new successor is also inactive, no changes in effectivity are necessary.
  - If its new successor's start date is unbounded, the rule's start date is made equal to its new successor's old start date, the rule's end date is made equal to its new successor's old end date, and the new successor's start and end dates are both made equal to its old end date.
  - If its new successor's start date is finite, the rule's start date and end date are both made equal to the new successor's start date.
- If the rule that you move is active (effective for some period of time), Configurator Developer:
  - Adjusts the start and end dates of the rule's old predecessor and successor to fill the gap created when you moved the rule. You can then modify these dates as required.
  - Adjusts the start and end dates of the rule that you moved to be consistent with its new context. For example:
    - * If its new predecessor is inactive, the rule that you moved becomes inactive.

    * If its new successor's effective dates range have specific start and end dates, both the start and end date of the rule that you moved are set to its predecessor's end date. You can then modify these dates as required.

To reorder rules in a Rule Sequence, see Section 30.16 on page 30-16.

To remove a rule from a Rule Sequence, see Section 30.17 on page 30-17.

Part IV

Runtime User Interfaces

Part IV describes how to create and maintain User Interfaces in Configurator Developer.

Part IV contains the following chapters:

19 Displaying the Model

This chapter presents general information about how a User Interface that you generate in Configurator Developer displays Model structure nodes and other data.

This chapter includes the following sections:

- Model Structure and Generated User Interfaces
- Refreshing a User Interface
- Controlling the Content of a User Interface
- Runtime Navigation
- The Configuration Summary Page
- Displaying Prices and Available to Promise Dates
- Multiple Language Support and the Runtime User Interface

19.1 Model Structure and Generated User Interfaces

When you generate a UI that is based on the Model's structure, the UI Master Template you select determines what types of UI elements are generated for each node in the Model. These UI elements include all option selection controls such as Check Boxes, Drop-down Lists, Radio Buttons, Item Selection Tables, and so on. UI templates are explained in Chapter 20, "User Interface Templates".

The Model's structure determines the initial structure of a generated UI, as well as the initial sequence in which UI Pages appear when viewing the UI structure in the User Interface area of the Workbench, and at runtime. The Model's structure also determines how UI elements are initially displayed on each UI Page.

For example, if Component A contains 5 Option Features, each Feature appears in Component A's UI Page in the same order as they appear in the Model structure: the UI control for the first child Feature in the structure appears at the top of the UI Page, followed by the second, and so on.

After generating a UI, you can modify the order and appearance of UI Pages and their content in the UI area of the Workbench. For details, see Chapter 31, "User Interface Area of the Workbench".

19.1.1 BOM and Non-BOM Model Structure

Model structure nodes generally fall into two categories: BOM nodes and non-BOM nodes (a non-BOM nodes is one you create in Configurator Developer). Therefore, all UI Master Templates have BOM Content and Non-BOM Content settings. These settings are described in Chapter 20, "User Interface Templates".

All Model structure that you create in Configurator Developer under a BOM Model appears before any child BOM nodes in the Model's structure. Therefore, when you generate a UI, any guided buying or selling content appears before the BOM content when viewing the UI in the User Interface area of the Workbench, and at runtime (for example, in a UI that provides step-by-step navigation). This behavior is usually preferable to displaying non-BOM content at the end of a UI, but you can modify the order in which UI content appears in the User Interface area of the Workbench, if necessary.

19.1.2 Model Structure and Effectivity

The effectivity defined within a Model affects whether nodes appear in the runtime UI (or the Model Debugger). For example:

- Nodes that are not available because of their effective date or the Usage(s) to which they are assigned do not appear in a runtime UI.
- Any links, buttons, images, and action buttons that control navigation to an ineffective (hidden) UI Page do not appear in a runtime UI. If you defined a Configurator Extension to navigate to a page but that page is not effective in the runtime UI, Oracle Configurator displays a message that tells the end user why it is not available.
- If a Model contains References to another Model and that Model is not effective, none of the referenced objects appear in the UI.

For more information, see Chapter 6, "Effectivity".

19.2 Refreshing a User Interface

While unit testing and debugging a configuration model and a UI, you must refresh the UI to ensure that any changes to the Model structure appear at runtime. When you refresh a UI, Configurator Developer refers to the latest version of the UI Master Template that was used to generate the UI and then adds, deletes, or modifies existing UI elements based on how the Model structure has changed.

You can use the Refresh Enabled setting to prevent an entire UI or specific UI elements from being refreshed. For details, see Section 19.2.4, "The Refresh Enabled Setting" on page 19-8.

You can refresh all of a Model's UIs in the General area of the Workbench, or refresh one UI at a time in the User Interface area of the Workbench. For details, see:

- Section 28.8, "UI Refresh Status" on page 28-6
- Section 31.3.11, "Refreshing a User Interface" on page 31-28

Refreshing a UI is required when the Model structure changes in:

- Configurator Developer
- Oracle Bills of Material - In this case, refreshing the BOM Model is required for the changes to appear in Configurator Developer.

If changes to the Model structure are minimal, refreshing the UI is sufficient. If the Model structure has changed extensively, Oracle recommends that you create a new UI. For details, see:

- Section 19.2.1, "Changes that Require a User Interface to be Refreshed" on page 19-3

- Section 19.2.2, "Changes that Do Not Require a User Interface to be Refreshed" on page 19-7

19.2.1 Changes that Require a User Interface to be Refreshed

This section lists the kinds of changes to the Model structure that require a User Interface to be refreshed, and describes what occurs when you refresh the UI.

A User Interface must be refreshed when:

- Nodes are Added to the Model
- Nodes are Moved within the Model
- Nodes are Deleted from the Model
- Nodes are Modified in Certain Ways

Refer to the following sections for details.

See also Section 19.2.2, "Changes that Do Not Require a User Interface to be Refreshed" on page 19-7.

19.2.1.1 Nodes are Added to the Model

You can create Model structure in:

- Configurator Developer, by manually creating nodes or copying existing nodes
- Oracle Bills of Material, by adding items to an imported BOM Model and then refreshing the BOM Model

A Model node's Display in User Interface setting also affects whether Configurator Developer creates UI elements for new nodes when you refresh the UI. For example, this setting is set to No (not selected) for Feature X when you create the UI. Therefore, no UI element is created for Feature X. You edit Feature X, select Display in User Interface, and then save the change. When you refresh the UI, Configurator Developer creates a new element for Feature X.

Conversely, if a UI element exists for a Model node and you change the node's Display in User Interface setting from Yes to No, Configurator Developer deletes its corresponding element when you refresh the UI.

19.2.1.1.1 Sequence of Model Structure and UI Elements after UI Refresh When you add nodes to the Model structure and then refresh a UI, Configurator Developer creates the new UI elements in the same order and location in which the nodes appear in the Model structure (this assumes that the Display in User Interface is set to Yes for the new nodes). In other words, Configurator Developer updates the UI so that the elements appear in the same location as they would in a new UI that is generated from the Model structure. This is also the case when you move nodes within the Model structure and then refresh the UI. See Section 19.2.1.2, "Nodes are Moved within the Model" on page 19-5.

You can change the order in which UI elements appear at runtime by editing the UI in the User Interface area of the Workbench.

19.2.1.1.2 UI Refresh and Maximum Elements Per Page When you create a UI that is based on the Model structure, a Model node with many children may be split into several UI Pages. This is controlled by the Maximum Number of Model Elements per Page setting in the UI Master Template.

When you add new nodes to the same parent node, the number of UI elements per UI Page may exceed the UI Master Template setting after you refresh the UI. For example, Component C1 has 75 Features, and the Maximum Number of Model Elements per Page setting in the UI Master Template is 25. When you generate the UI, Configurator Developer splits C1 into three UI Pages with 25 UI controls (one for each Feature) on each Page. You then add 3 Text Features to Component C1 and then refresh the UI. Component C1's last UI Page now contains 28 UI elements.

If you reorder C1's children (the Text Features) before refreshing the UI, the UI elements created when you refresh the UI may appear on C1's first, second, or third UI Page. See Section 19.2.1.1.1, "Sequence of Model Structure and UI Elements after UI Refresh" on page 19-3.

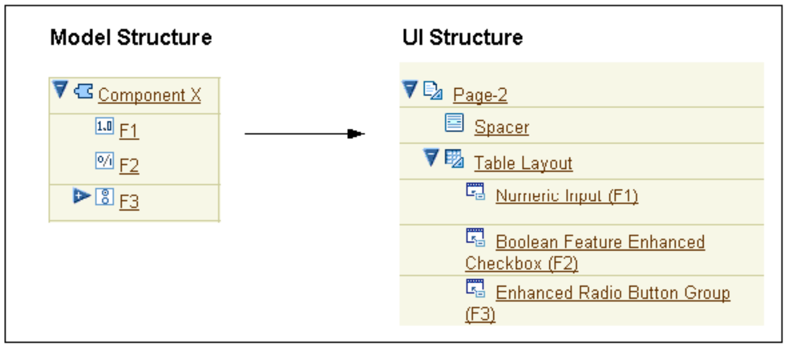
19.2.1.1.3 Examples of Adding Model Structure and then Refreshing a UI

Example 19–1 Creating Model Structure

1. In Model A, Component X has three children: F1, F2, and F3.
2. You generate a UI that is based on the Model structure.

   Configurator Developer generates a UI Page for Component X and UI elements for each of Component X's children. In the UI structure, the child elements appear under the Table Layout Region as shown in Figure 19–1.

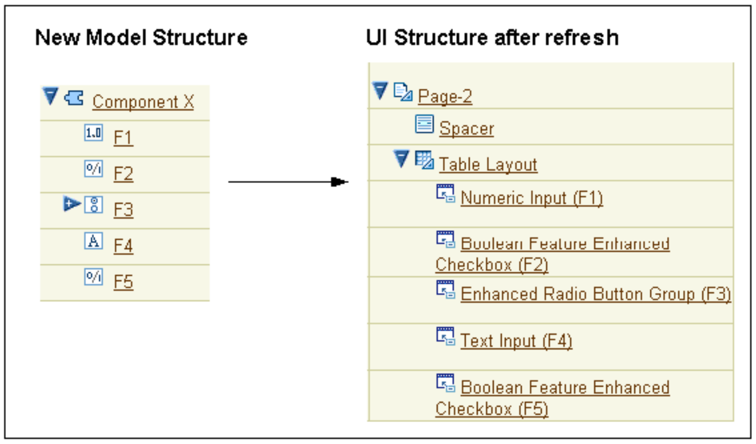
Figure 19–1 User Interface Generated from Model Structure



3. In the Model structure, you add Features F4 and F5 to Component X.
4. You refresh the UI.

   Using the same UI Master Template that was used to create the UI, Configurator Developer creates new UI elements for each new Feature and adds them as children of the Table Layout Region that appears under Component X's UI Page. See Figure 19–2 on page 19-5.

Figure 19–2 User Interface after UI Refresh


New Model Structure
UI Structure after refresh
Component X
F1
F2
F3
F4
F5
Page-2
Spacer
Table Layout
Numeric Input (F1)
Boolean Feature Enhanced Checkbox (F2)
Enhanced Radio Button Group (F3)
Text Input (F4)
Boolean Feature Enhanced Checkbox (F5)

Example 19–2 Creating UI Content for Nested Components

1. In a UI that is based on the Model structure, Page 1 exists for Component C1.
2. In the UI structure, you create a UI Page called Page 2 and specify Component C1 as its Associated Model Node.

   The Associated Model Node setting is explained in Section 21.14, "User Interface Elements and Associated Model Nodes" on page 21-43.
3. In the Model structure, you create Component C2 as a child of Component C1, and create several Features beneath C1.
4. You refresh the UI.

If the UI Master Template specifies that nested components are added to their parent's Page, Configurator Developer adds the elements that represent Component C2 and its children to Page 1.

If the UI Master Template setting specifies that new Pages are created for nested Components, then Configurator Developer generates a UI Page for Component C2. This Page follows Page 1 in the UI.

The UI Master Template setting referred to in the previous paragraphs is described in Section 20.2.2.2.1, "Defining Custom Pagination and Layout" on page 20-5.

For more information, see Section 19.2.3, "How the UI Master Template is used when Refreshing a User Interface" on page 19-7.

19.2.1.2 Nodes are Moved within the Model

When you move a node from one location in the Model structure to another, or change the order of nodes beneath their parent, you must refresh the UI. When you refresh the UI, Configurator Developer updates the UI structure so it reflects the Model structure.

Consider the following examples.

Example 19–3 Moving a Node that is Associated with a UI Page

1. You move a node that is associated with a UI Page (for example, a Component) from one location in the Model to another.
2. You refresh the UI.

Configurator Developer updates the navigation Menu, Page Links, and the UI Page element as necessary to reflect the node's new location in the Model structure.

Example 19–4 Moving Model Structure

1. You create a Check Box element and associate it with a Boolean Feature.
2. You move the Boolean Feature to a different parent in the Model structure (for example, from Component C1 to Component C2).
3. You refresh the UI.

If the Check Box's new parent element is refresh-enabled, Configurator Developer moves the Check Box from its previous location in the UI to its parent's UI Page. If the Check Box's new parent element is not refresh-enabled, the Check Box remains in its current location in the UI.

19.2.1.3 Nodes are Deleted from the Model

When you delete a UI element's associated Model node and then refresh the UI:

- If refresh is enabled for the element, Configurator Developer deletes the element and any of its children.

  See Section 19.2.4.2, "Refresh Enabled Setting: UI Element Level" on page 19-8.
- If refresh is disabled for the element, the element remains in the UI but it is disassociated from its Model node and is read-only at runtime. In other words, its Associated Model Node setting becomes null.

A UI element that has no associated Model node appears at runtime, but it is read-only. For example, an Oracle Configurator end user cannot use an Enhanced Check Box element to select an option at runtime if it has no associated Model node. You can modify the element by specifying a new associated Model node, or delete the element if you do not want it to appear in the UI.

Configurator Developer also disassociates an element from its associated Model node when you refresh the UI after the node is moved out of the UI Page's scope. For details, see Section 21.15, "Associated Model Nodes and Page Scope" on page 21-44.

19.2.1.4 Nodes are Modified in Certain Ways

Changes to Model nodes that occur in Configurator Developer and require a UI to be refreshed include:

- The Display in User Interface setting changed
- The Enable Option Quantities setting changed on an Option Feature
- The Maximum Selections value changed to or from '1' on an Option Feature
- The Instantiability setting on a Model Reference or Component changed. For example:
  - From Required Single Instance to Optional Single Instance
  - From Optional Single Instance to Required Single Instance
  - From Multiple or Variable Instances to either Required Single Instance or Optional Single Instance

Changes to Model nodes that occur in Oracle Bills of Material and require a UI to be refreshed (after the BOM Model is refreshed) include:

- The Maximum Quantity changed to or from '1' on any BOM node

- The Mutually Exclusive setting changed for a BOM Model or BOM Option Class

All of the changes listed in this section (except the one related to the Display in User Interface setting) cause a different template to be used to display the modified node when you refresh the UI. For details, see:

- Section 19.2.5, "UI Template References and UI Refresh" on page 19-9
- Section 19.2.6, "Elements Created from a UI Content Template and UI Refresh" on page 19-9

19.2.2 Changes that Do Not Require a User Interface to be Refreshed

If the Model structure has not changed in any way since you generated or refreshed a UI, you do not have to refresh it. You also do not have to refresh a UI after editing it to see your changes at runtime. Editing a UI is described in Section 31.3, "Editing a User Interface" on page 31-2.

Refreshing a UI is also not required if a UI Content Template that the UI refers to has changed. A UI that references a UI Content Template dynamically renders the template's content at runtime, so any changes to the template appear automatically. For more information about UI templates, see Chapter 20.

19.2.3 How the UI Master Template is used when Refreshing a User Interface

Refreshing a UI creates new UI elements or modifies existing elements based on the latest version of the UI Master Template that was used to create the UI. Therefore, changes to the Master Template that occur since you created the UI will be reflected in any new UI content after you refresh the UI.

Refreshing the UI also updates elements for nodes that have changed since the UI was generated or refreshed and require a different template to display them at runtime. For details, see:

- Section 19.2.5, "UI Template References and UI Refresh" on page 19-9
- Section 19.2.6, "Elements Created from a UI Content Template and UI Refresh" on page 19-9

When you refresh a UI, Configurator Developer checks the UI Master Template to determine:

- Which UI Content Template to use to display any new or modified nodes
- Whether Configurator Developer copies UI Content Templates as page content (UI elements), or incorporates them into the UI by creating Template References

  The Template Usage setting is described in Section 20.3.1, "Specifying How a User Interface Uses Content Templates" on page 20-14.

If the UI Master Template specifies that a UI element is created as page content, Configurator Developer creates a new UI element to replace the existing element when you refresh the UI. For example, the Maximum Selections for an Option Feature changes from 3 to 1. When you refresh the UI, the Enhanced Checkbox Group for that node is replaced with a Enhanced Radio Button Group.

If the Master Template specifies that the template should be incorporated by reference into the UI, the resulting Template Reference refers to a different UI Content Template after the UI is refreshed. For example, a Template Reference that is associated with a BOM Option Class refers to the Multi-Select BOM Item Table template. The Mutually Exclusive setting for this BOM node changes from No to Yes in Oracle Bills of Material, and the BOM Model is refreshed. When you refresh the UI, the Template Reference now refers to the Single-Select BOM Item Table template.

19.2.4 The Refresh Enabled Setting

The Refresh Enabled setting is available in the details page for the UI Definition and specific UI elements. For details, see:

- Section 19.2.4.1, "Refresh Enabled Setting: User Interface Level" on page 19-8
- Section 19.2.4.2, "Refresh Enabled Setting: UI Element Level" on page 19-8

19.2.4.1 Refresh Enabled Setting: User Interface Level

The Refresh Enabled setting in the UI Definition's details page controls whether Configurator Developer checks the UI to see if it must be refreshed before the Model can be unit tested or published. The UI Definition is explained in Section 21.3 on page 21-3.

If UI refresh is disabled in the UI Definition details page, the UI does not appear in the General area of the Workbench as a UI that must be refreshed, regardless of whether the Model structure has changed. See Section 28.8, "UI Refresh Status" on page 28-6. Therefore, the UI is not refreshed when you refresh all of a Model's UIs from the General area of the Workbench. However, you can override this setting and refresh the UI from the User Interface area of the Workbench at any time.

By default, UI refresh is:

- Enabled for UIs that are generated from the Model structure
- Disabled for empty UIs (that is, UIs that you create from scratch in the User Interface area of the Workbench)

19.2.4.2 Refresh Enabled Setting: UI Element Level

You may want to disable UI refresh on a specific element if you want to maintain the element and want to prevent Configurator Developer from overriding any of your changes.

For example, you may create a region that displays Model content in a highly customized way, or you may want to prevent Configurator Developer from removing or updating a specific UI Template Reference when you refresh a UI. For example, a Model Reference's instantiability changes from Required Single Instance to Multiple or Variable Instances. If you want the UI Template Reference to continue to refer to the same template, deselect the Refresh Enabled setting in the element's details page.

If you deselect Refresh Enabled for a UI element, Configurator Developer does not update the element or any of its children when you refresh the UI. This is true even if refresh is enabled for any of the element's children. In other words, disabling refresh for a parent element's takes precedence over the Refresh Enabled setting for any of its children.

The Refresh Enabled setting appears in the details page of the following UI elements:

- UI Pages
- Layout Regions
- Template References
- Page Flows
- Menus

For details about these elements, see Chapter 21, "User Interface Structure and Design".

When you create a UI that is generated from the Model structure, UI refresh is enabled for all UI elements by default.

19.2.5 UI Template References and UI Refresh

For background information about UI Template References, see Section 21.16.1, "User Interface Template References" on page 21-48.

It is possible for the target of a Template Reference to change to a different UI Content Template when you refresh the UI. This occurs only when the Template Reference's associated Model node has changed such that the template that was originally the Template Reference's target is no longer appropriate for displaying the node at runtime.

For example, the Template Reference that is associated with an Option Feature has the Dynamic Drop-down Control template as its target. You change the Feature's Maximum Selections value from 1 to 3, and then refresh the UI. Configurator Developer checks the UI Master Template and changes the Template Reference's target to the Enhanced Checkbox Group template. This is because the Option Feature now corresponds to the "Multi- Select" setting in the UI Master Template, and continuing to display the node using a drop-down control would allow end users to select only one of the Feature's Options at runtime.

The following changes do *not* cause the target of a Template Reference to change when you refresh a UI (these changes occur when editing a UI):

- You change a Template Reference's target (that is, you specify a different UI Content Template)

  In this case, refreshing the UI does not change the target back to the original template, even if the new target does not match the template specified for the Model node in the UI Master Template.

- You change a Template Reference's Associated Model Node setting (that is, you specify a different Model node)

  In this case, refreshing the UI does not change the target to a different template, even if the new node's type does not match the template specified for that node in the UI Master Template.

Configurator Developer does not "undo" your UI edits when refreshing a UI. In other words, when you refresh a UI, Configurator Developer updates the UI to reflect changes you make to the Model structure, not changes you make when editing the UI. For details, see Section 19.2.1, "Changes that Require a User Interface to be Refreshed" on page 19-3.

19.2.6 Elements Created from a UI Content Template and UI Refresh

Before reading this section, you should understand how UI elements are created from UI Content Templates. For details, see Section 21.16.2, "Creating UI Content from a User Interface Content Template" on page 21-50.

Refreshing a UI replaces existing UI content that was created using a UI Content Template only if both of the following are true:

- The node specified as the Associated Model Node for the region's root node was modified

- The modified node now corresponds to a different setting (and template) in the UI Master Template

In this case, Configurator Developer replaces the existing content with the content of the UI Content Template that is specified by the UI Master Template for the node's new type.

For example, you create UI content using the Multi-Select BOM Item Table with Header UI Content Template and select a BOM Option Class as the root element's Associated Model Node (the root element for the Multi-Select BOM Item Table is a Flow Layout). You then edit the root element (the Flow Layout), enable UI refresh, and set its Associated Model Node to an Option Feature that has a Maximum Selections value of 3. When you refresh the UI, Configurator Developer replaces the UI content with the content of the Enhanced Checkbox Group UI Content Template (since this is the template specified for the Multi-Select Option Features setting in the UI Master Template).

If you delete the root element's associated Model node, then Configurator Developer deletes the UI element and its children (that is, the entire region) when you refresh the UI. If you move the root element's associated Model node, then Configurator Developer moves the element and its children to reflect the node's new location in the Model structure when you refresh the UI. See Section 19.2.1.1.1, "Sequence of Model Structure and UI Elements after UI Refresh" on page 19-3.

If you delete or move the root element's associated Model node, and the root element is not refresh-enabled, then Configurator Developer disassociates the element and the Model node when you refresh the UI. In this case, the UI element will be read-only at runtime and you may want to delete the element or assign it to a different Model node.

19.3 Controlling the Content of a User Interface

There may be portions of a Model's structure that you do not want to display in the User Interface. For example, you define a Component that is a collection of Totals used in calculations required by the configuration. You use these nodes when testing your runtime Configurator, but do not want your end users to see them.

To prevent Configurator Developer from generating a UI element for a node or its children, open the node for editing in the Structure area of the Workbench, and then deselect Display in User Interface. This ensures that Configurator Developer does not generate UI elements for the node or any of its children when you generate or refresh the UI.

You can also dynamically hide or show UI elements at runtime based on conditions that you define. This is explained in Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

A UI element does not appear in the UI if its associated Model node is not effective at runtime. For more information, see Chapter 19.1.2, "Model Structure and Effectivity" on page 19-2.

User Interface templates determine the default content of a UI that is based on the Model structure. For details, see Chapter 20, "User Interface Templates".

19.4 Runtime Navigation

The UI Master Template you select when generating a UI determines the UI's primary navigation method. The available methods include step-by-step, side menu, subtab, and Model tree. For details, see the description of each predefined UI Master Template in Section 20.2, "User Interface Master Templates" on page 20-2.

You can also provide additional navigation capabilities by creating navigation buttons or links to UI Pages. Refer to the following sections for more information:

- Section 31.3.7, "Creating a Page Link" on page 31-26
- Section 31.3.4.18, "Creating a Custom Button" on page 31-18

At runtime, the primary navigation controls available in a UI may change when a referenced Model's navigation style is different from the parent Model's. See Example 19–5 on page 19-11.

Example 19–5 Alternate Navigation Defined in a Referenced Model's UI

The generated UI for a Model uses the Single Level Menu primary navigation style. The UI for one of its child (referenced) Models was generated using a different UI Master Template that uses the step-by-step navigation style. At runtime, when an end user navigates into the child Model (for example, by using a drilldown control), the first Page in the Page Flow is displayed and the Menu for navigating through the parent Model is replaced by a set of Back and Next buttons for navigating through the child Model. After configuring the child Model, the end user clicks either a Cancel button or an Apply button to return to the parent Model. Clicking Apply saves all of the changes made in the child Model; clicking Cancel loses the changes.

In this example, clicking the Back button only displays the previous page in the flow; it does not enable the end user to return to the parent Model.

For more information, see Section 4.4.1, "Integrating Referenced User Interfaces" on page 4-3.

19.4.1 Using a Web Browser's Forward and Back Controls

The runtime Oracle Configurator supports the use of a Web browser's Forward and Back navigation controls. However, it is recommended that end users use the navigation controls provided in the generated UI, such as the navigation tree, a menu of Page Links, or Back and Next buttons (in a UI that provides step-by-step navigation).

When an Oracle Configurator end user navigates using the browser's Back or Forward button, a warning message indicates that the page data has expired. At this point, the end user must resubmit the data (for example, by clicking Refresh), and then confirm the action to continue.

19.5 The Configuration Summary Page

The Configuration Summary page lists all items selected during a configuration session. An Oracle Configurator end user can navigate to this page at any time to, for example, review all selections when configuring an item, or at the end of a session before saving the configuration. The predefined UI Master Templates display Cancel, Return to Configuration, and Finish buttons on this page by default.

When you generate a UI that is based on the Model's structure, the predefined UI Master Template you select displays a Preview Configuration button on each UI Page at runtime. You can also enable end users to navigate to the Configuration Summary page by creating a UI control and assigning the Go to Preview Page action to it. For details, see Section 21.13, "User Interface Actions" on page 21-38.

By default, all Items in the CZ schema's Item Master that have the Orderable check box selected appear in the Summary page. For details about this setting, see Section 2.3, "Orderable Items" on page 2-2.

The predefined Summary Page UI Content Templates are described in Section 20.3.5.2 on page 20-22.

19.6 Displaying Prices and Available to Promise Dates

By default, pricing and Available to Promise (ATP) information does not appear in a runtime Oracle Configurator launched from a host application or when unit testing a configuration model from Configurator Developer (in other words, using the Model Debugger or a generated UI). To enable pricing or ATP in any of these environments, you must first define the Oracle Configurator Servlet property `cz.activemodel`. This property and others are described in the *Oracle Configurator Installation Guide*.

If pricing and ATP are enabled, some additional setup is required to display pricing and Available to Promise (ATP) information when unit testing. For details, see Section 22.4, "Displaying Pricing Information and ATP Dates when Unit Testing" on page 22-3.

For more information about displaying pricing and ATP in a runtime Oracle Configurator, see the *Oracle Configurator Implementation Guide*.

For important information about displaying pricing and ATP using UI Content Templates, see Section 20.5 on page 20-23.

19.7 Multiple Language Support and the Runtime User Interface

If you have implemented Multiple Language Support (MLS), you can create a single UI to use with multiple languages, or create a different UI for each language in which you do business.

For more information, see Appendix B, "Multiple Language Support".

20 User Interface Templates

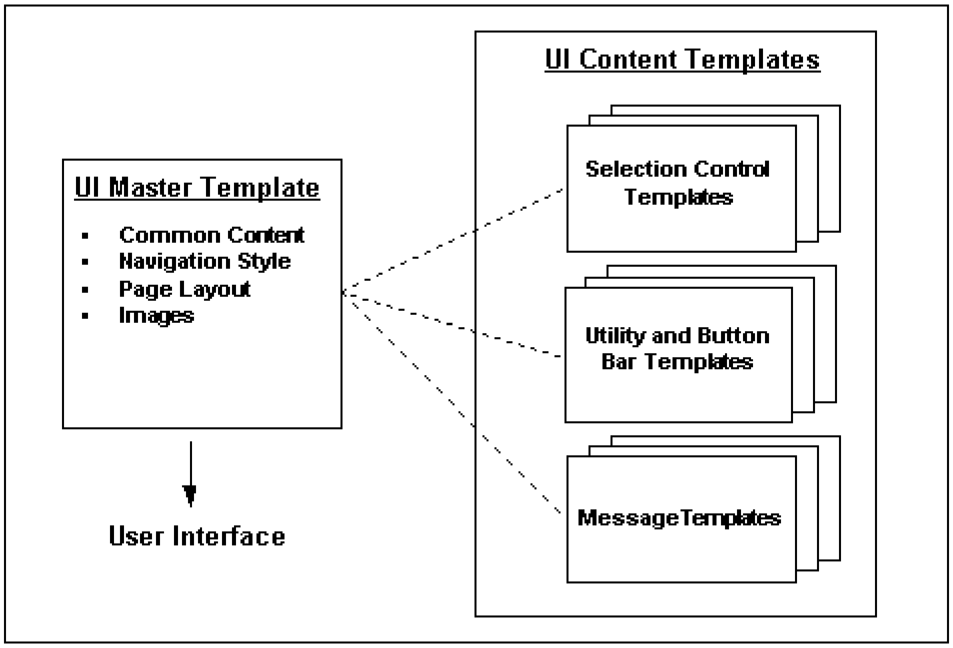
This chapter describes the templates used to generate a User Interface in Configurator Developer.

This chapter includes the following sections:

20.1 Introduction

There are two types of User Interface templates: User Interface Master Templates and User Interface Content Templates. A UI Master Template consists of various settings that are used when you generate a User Interface, and specifies which UI Content Templates are used to display Model content, images, messages, and so on. See Figure 20–1 on page 20-2.

Figure 20–1 User Interface Templates



Configurator Developer provides predefined UI Master Templates and UI Content Templates. These templates are located in the UI Templates Folder in the Main area of the Repository. All predefined UI templates are read-only and cannot be modified or deleted. However, if a predefined template does not meet your needs, you can either create a copy of it and modify it, or create a new template from scratch.

For details about each type of template, see:

- Section 20.2, "User Interface Master Templates" on page 20-2
- Section 20.3, "User Interface Content Templates" on page 20-13

20.2 User Interface Master Templates

A UI Master Template consists of settings that control:

- The UI's default structure, pagination, navigation method, and layout
- Which UI Content Templates display UI elements for Model structure nodes
- How page-level buttons appear at runtime
- How the Configuration Summary page appears at runtime
- How required and optional messages appear at runtime
- The images to use at runtime to indicate the state of all options

See Figure 20–1, "User Interface Templates" on page 20-2.

You must select a UI Master Template when generating a UI that is based on the Model structure, or when generating an empty UI. When you generate a UI that is based on the Model structure, Configurator Developer generates UI elements for Model structure nodes using the UI Content Templates specified by your UI Master Template. The Master Template also determines the Message Templates, Utility Templates, and images that are used at runtime.

When you generate an empty UI, the UI Master Template you select determines only which Message Templates, Utility Templates, and images are used at runtime. You must define the UI's content, including UI Pages, selection and navigation controls, and so on. This is described in Section 31.3, "Editing a User Interface" on page 31-2.

> **Tip:** Plan to spend time designing and customizing UI templates to produce the optimal User Interfaces for your configuration models and your end users. For example, generate UIs using the predefined UI Master Templates and review them at runtime. Then experiment with creating your own templates and customizing them until you are satisfied with the UI's appearance and behavior. When your UI Master Templates are complete, they can easily be reused to generate UIs that meet your specific requirements.

For more information, see:

- Section 25.3.7, "Creating a User Interface Master Template" on page 25-5
- Section 31.4, "Creating a User Interface Content Template" on page 31-30
- Section 20.3, "User Interface Content Templates" on page 20-13

20.2.1 Default Settings for the Predefined User Interface Master Templates

Configurator Developer provides the following predefined UI Master Templates:

- Oracle Browser Look and Feel with Step-by-Step Navigation UI Master Template on page 20-10
- Oracle Browser Look and Feel with Dynamic Tree Navigation UI Master Template on page 20-11

You can also manually create the following UI Master Templates:

- Step-by-Step Navigation UI Master Template on page 20-11
- Dynamic Model Tree Navigation UI Master Template on page 20-11
- Single-Level Side Navigation UI Master Template on page 20-12
- Multiple-Level Side Navigation UI Master Template on page 20-12
- Subtab Navigation UI Master Template on page 20-12
- Single Page Layout UI Master Template on page 20-13

The predefined UI Master Templates and any Master Template that you create have default values for each setting, such as which UI Content Templates are used to display controls for selecting options, runtime messages, images, and so on.

The only difference between the predefined UI Master Templates is the navigation style that each provides. All of the other settings are the same, except where indicated in Table 20–1 on page 20-3.

Table 20–1 Default Settings for the Predefined UI Master Templates

Setting	Default Value
Primary Navigation	Step-by -Step or Dynamic Model Tree
Show Train for all Multi-Page Flows	Yes, for Step-by-Step template, No for Dynamic Model Tree
Pagination	More, smaller pages

Table 20–1 (Cont.) Default Settings for the Predefined UI Master Templates

Setting	Default Value
Nested Option Classes	Add to Parent Page
Nested Components	Add to Parent Page
Referenced Models	Create New Drilldown Pages
Maximum Number of Model Elements per Page	100
Control Type (Drilldown Controls region)	Button
Control Text (Drilldown Controls region)	Configure &DISPLAYNAME &DISPLAYNAME refers to the associated node's DisplayName System Property. For details, see Section 5.3, "System Properties" on page 5-2.)
Template Usage (for all UI Content Templates)	Incorporate by Reference (see Section 20.3.1 on page 20-14)

For more information about these settings, see Section 20.2.2, "UI Master Template Information and Settings" on page 20-4.

20.2.1.1 Default Pagination Settings

When choosing a Pagination setting, you can select either "Fewer, larger pages" or "More, smaller pages." These settings determine the default values for the detailed pagination settings, which are visible in the Custom Pagination page. See Table 20–2 on page 20-4.

Table 20–2 Default Pagination Settings

High-Level Setting	Detailed Settings (Custom Pagination page)
Fewer, larger pages	Maximum Number of Model Elements per Page = 100 Nested Option Classes = Add to Parent Page Nested Components = Add to Parent Page Referenced Models = Create New Drilldown Pages Number of Rows Shown in Tables = 25
More, smaller pages	Maximum Number of Model Elements per Page = 30 Nested Option Classes = Create New Pages in Primary Navigation Nested Components = Create New Pages in Primary Navigation Referenced Models = Create New Drilldown Pages Number of Rows Shown in Tables = 25

For more information, see Section 20.2.2.2.1, "Defining Custom Pagination and Layout" on page 20-5.

20.2.2 UI Master Template Information and Settings

This section describes information and settings that are common to all UI Master Templates. You access these settings by clicking links that appear in the left-hand side of each page in a UI Master Template.

When viewing or editing a UI Master Template, the UI Content Template that it refers to are displayed with an icon that displays the path to the template in the Main area of the Repository. For example, to view the Folder in which a specific template resides, place the cursor over the icon or press the Tab key until the path information appears.

All UI Master Templates contain the following sections:

- "General Section" on page 20-5
- "Pagination and Layout Section" on page 20-5
- "BOM Content Section" on page 20-7
- "Non-BOM Content Section" on page 20-8
- "Utility Templates Section" on page 20-8
- "Message Templates Section" on page 20-8
- "Images Section" on page 20-9

20.2.2.1 General Section

These settings include the name, description, and any notes about the UI Master Template.

20.2.2.2 Pagination and Layout Section

These settings define the overall structure and organization of the UI.

The Pagination and Layout page contains the following settings:

- **Primary Navigation**: This indicates the type of navigation that the UI Master Template provides at runtime. For details about each type, refer to the descriptions of each UI Master Template in Section 20.2 on page 20-2.

  For details about primary navigation methods in referenced UIs, see Section 19.4, "Runtime Navigation" on page 19-10.

  To display a progress indicator at runtime when an end user is performing a task that consists of three or more steps, select Show Train for all Multi-Page Flows.

- **Pagination**: Specify whether you want the UI to have more pages with fewer options per page (More, smaller pages), or fewer pages with more options on each page (Fewer, larger pages). For more information, see Table 20–2 on page 20-4.

  For greater control over the pagination, select Custom Settings, and then click Define. See Section 20.2.2.2.1, "Defining Custom Pagination and Layout" on page 20-5.

- **Page Status Area Template**: Specifies which template Configurator Developer uses to generate the content of the page status area. Section 20.3.5.3, "Icon Legend Template" on page 20-22.

20.2.2.2.1 Defining Custom Pagination and Layout Following are the settings that are available when you choose to define custom pagination and layout settings for a UI Master Template. That is, when editing a UI Master Template, you go to the Pagination and Layout section, select Custom Settings, and then click Define.

Pagination of Model Structure Section

The settings in this section specify how end users access "nested" configurable components at runtime. Nested configurable components include BOM Option Classes, nested Components, and referenced Models. A nested component results when a node contains another node of the same type, such as a Component within another Component. You can specify whether you want to display the selection control for a nested component on the same page as its parent, require the end user to "drill down" into a nested transaction, or create a separate UI Page for each nested component.

For example, if you want a UI control that represents a BOM Option Class to appear on its parent node’s UI Page (for example, a BOM Item Selection Table), select Add to Parent Page. To create a new Page that end users access using the UI’s primary navigation method (such as Step-by-Step), select Create New Pages in Primary Navigation.

To create new pages that end users access by clicking a button or other navigation control, select Create New Drilldown Pages. See "Drilldown Controls Section" on page 20-6.

Note: When the UI Master Template’s Primary Navigation is anything other than “Single Page,” Configurator Developer generates new UI Pages for the Model root and its first-level child Components and Option Classes. Configurator Developer uses the Pagination of Model Structure settings to determine how to generate content for nested Components and Option Classes that exist below the first-level children.

The Number of Model Elements per Page setting limits how many UI controls can appear in the same UI Page. Examples of UI controls include list boxes, drop-down lists, input fields, and so on. When a table contains UI controls, each row counts toward the maximum allowed per page. Limiting how many elements appear on each page may improves the usability and performance of a User Interface.

For example, your Model contains 100 Option Features within Component X and each Feature contains many Options. You do not want to display all of the Features on a single UI Page, because end users may find it difficult to make selections. If you specify 10 Model elements per page, Configurator Developer creates ten UI Pages for Component X and displays 10 Features on each page. If your UI Master Template displays a Menu, the runtime UI displays a separate Page Link for each UI Page and numbers the Page Links sequentially (for example, Component X (1), Component X (2), and so on).

The Number of Rows Shown in Tables setting determines how many rows are displayed in a tabular UI control, such as a BOM Item Table. (BOM Item Tables are described in Section 20.3.3.1 on page 20-17.) If the number of rows in a table is greater than this limit, the UI Page displays Previous and Next links the end user can select to display additional rows.

Drilldown Controls Section

These settings specify the type of controls that provide drilldown navigation into a nested transaction, such as when configuring a referenced Model, a non-instantiable subcomponent, or a BOM Option Class.

Select a Control Type to specify whether a Button, Link, or Image is provided to enable end users to access nested UI structure. For example, Model A contains a Reference to Model B. The UI Page for Model A must contain a control to enable the end us to navigate to Model B.

The text you enter in the Control Text field appears as the Button label, the text of the link, or the text that appears when the end user places the cursor over the image at runtime. Where the text appears depends on the Control Type selected.

If the Control Type is Image, enter the path and file name of the image to display in the Image URL field. For example, `/OA_MEDIA/drilldown_control.gif`. To appear at runtime, the file must exist on the server on which Oracle Configurator is running.

20.2.2.3 BOM Content Section

The BOM UI Layouts setting controls how BOM Models and BOM Option Classes appear in the UI. Select Standard Layout if you want all BOM Option Classes and BOM Standard Items to appear in BOM Item Tables. Select Compact Layout to display BOM Option Classes and BOM Standard Items as drop-down lists, check box groups, and radio button groups.

Which control appears depends on whether an end user can enter a quantity at runtime, and whether the node is:

- A BOM Standard Item
- Mutually exclusive
- Nested BOM structure

For a definition of nested structure, see Section 20.2.2.2.1 on page 20-5.

> **Note:** Pricing and ATP information does not appear if you select Compact Layout, even if pricing and ATP are enabled.

To specify different settings, select Custom Settings, and then click Define (see below).

20.2.2.3.1 Customizing the Display of BOM Content Use the settings in the BOM Content Custom Settings page to change the default UI Content Templates that Configurator Developer uses to generate BOM content. You access these settings when defining or editing a UI Master Template by navigating to the BOM Content page, selecting Custom Settings, and then clicking Define.

For each setting, a BOM Selection Control UI template determines which type of UI control Configurator Developer generates for each BOM item in your Model. For a description of these templates, see Section 20.3.3 on page 20-16.

Specify a BOM Control Template Usage to control how each UI Content Template is incorporated into the generated UI. For details, see Section 20.3.1 on page 20-14.

For each type of BOM structure listed in the Custom Settings page, specify the UI Control Template you want to use for:

- BOM Standard Items that are mutually exclusive or not

  For a definition of mutually exclusive, see Section 11.3 on page 11-3.
- BOM Standard Items for which entering a quantity at runtime is required or optional (see Note below)
- Nested BOM structure for which entering a quantity at runtime is required or optional (see Note below)

Note: "Quantities Required" and "Quantities Optional" refers to whether a UI control that accepts a quantity for the BOM Item is provided at runtime. By default, a quantity input control is provided if the Maximum Quantity setting in Oracle Bills of Material for the Item is greater than 1. If the Maximum Quantity is less than 1, a quantity input control does not appear.

In either case, you can choose a UI Control Template that provides a quantity input control, or one that does not.

- Nested BOM structure (such as a BOM Option Class) that is mutually exclusive or not

  For a definition of nested structure, see Section 20.2.2.2.1 on page 20-5.
- BOM Model References that can have multiple instances or a single, optional instance

20.2.2.4 Non-BOM Content Section

Use the settings in this page to specify how Model structure created in Configurator Developer appears in the UI. These nodes include Components, Features, non-BOM Model References, Totals, Resources, and Connectors. Select Standard Layout to display single-select Option Features (min/max = 1/1) as radio button groups. Select Compact Layout to display them as drop-down lists.

To specify different settings, select Custom Settings, and then click Define (see below).

20.2.2.4.1 Customizing the Display of Non-BOM Content Use the settings in the Non-BOM Content Custom Settings page to change the default UI Content Templates that Configurator Developer uses to generate non-BOM content in the runtime UI.

Specify a Non-BOM Control Template Usage to determine how each UI Content Template is incorporated into the generated UI. For details, see Section 20.3.1 on page 20-14.

Specify the UI Control Template you want to use to display:

- Option Features for which only one Option or multiple Options can be selected, with or without a quantity (see Section 20.3.3.6 on page 20-19)
- Instantiable Components and References to non-imported Models that are either optional with a single instance, or may have multiple instances (see Section 20.3.3.3 on page 20-17)
- Other content, including Totals, Resources, Connectors, Boolean Features, Text Features, and Numeric Features (see Section 20.3.3 on page 20-16)

20.2.2.5 Utility Templates Section

In this section, specify which templates are used to display common UI content, such as button bars, connection choosers, and the Configuration Summary page.

In the Button Bar Templates section, select templates that control the type of button bar displayed for the following:

- **Basic Transaction**: Specifies the template to use for both top-level and nested transactions (processes) when not in a step-by-step flow. For a definition of nested structure, see Section 20.2.2.2.1 on page 20-5.

- **Two-Page Flow Navigation**: Specifies the template to use for step-by-step navigation when there are only two steps in a task flow.
- **Multi-Page Flow Navigation**: Specifies the template to use for task flows that consist of more than two pages.

The predefined Navigation Button Bar Templates are described in Section 20.3.2 on page 20-15.

In the Utility Page Templates section, specify the template to use for the Standard Configuration Summary Page, Configuration Upgrades Summary Page (Filtered and Complete), and the Connection Chooser page.

The predefined Summary Page Templates are described in Section 20.3.5.2 on page 20-22. The predefined Connection Chooser template is described in Section 20.3.3.12 on page 20-19.

20.2.2.6 Message Templates Section

Use the settings in this page to specify the templates to use with various required and optional runtime messages. For each optional Message Template, either specify the template to use, or click Clear if you do not want to display a message at runtime. For example, if you do not want end users to see a confirmation message when deleting a component at runtime, be sure no template is specified for the Confirm Delete Instance setting.

After generating the UI, you can modify which Message Templates are used at runtime. For details, see Section 31.3.1 on page 31-3.

The predefined Message Templates are described in Section 20.3.4 on page 20-20.

20.2.2.7 Images Section

These settings specify the images to display in a runtime User Interface for Enhanced Check Boxes, Enhanced Radio Buttons, and Status Indicator images. Each setting has a default file name and absolute path.

You can override the images specified here after generating a UI. For details, see Section 31.3.1 on page 31-3.

The images that Configurator Developer provides are located in the `OA_MEDIA` directory. For example:

```
/OA_MEDIA/chkbox_userselected.gif
```

You can optionally use different images for each setting. For example, you may want to display a red "X" next to options that the system has excluded from the configuration, or a green asterisk next to options that contain required selections. Note that all images that you specify in this page must be 16x16 pixels.

If you specify a different image file that is stored in a subdirectory within `OA_MEDIA`, enter the file name with a path relative to `OA_MEDIA`. For example:

```
/OA_MEDIA/MySubdirectory/ImageName.gif
```

You can also specify a complete path to an image. For example:

```
http://www.MyWebSite.com/Images/Image001.gif
```

Note: The images that appear in a referenced Model's UI may be different than those used in the parent Model's UI. For details, see Section 4.4.1, "Integrating Referenced User Interfaces" on page 4-3.

20.2.2.7.1 Enhanced Check Box and Radio Button Images If your UI Master Template uses the Enhanced Radio Button Group or Enhanced Check Box Group UI Control Templates, these settings control which images are used to indicate an option's selection state at runtime. The default images are shown in Figure 20–2 on page 20-10.

For more information, see Section 20.3.3.6, "Enhanced Check Box Group and Enhanced Radio Button Group Control Templates" on page 20-19.

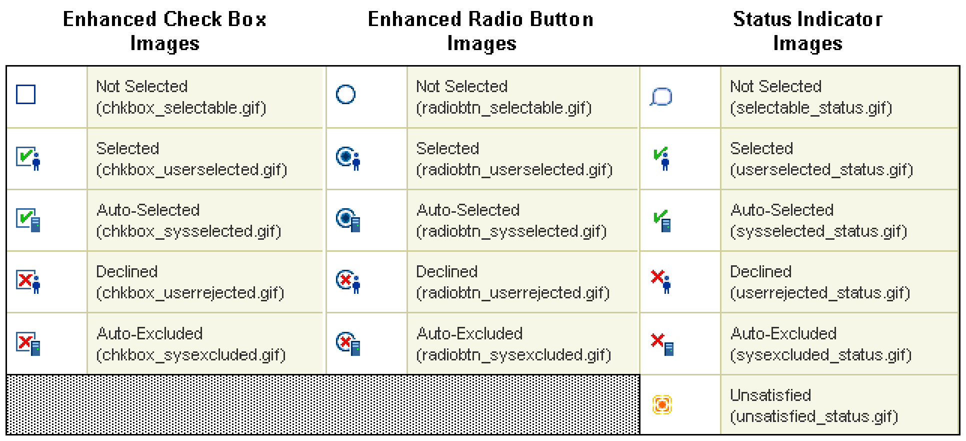
20.2.2.7.2 Status Indicator Images The runtime UI uses the images specified in this section to indicate the selection state of options that appear as standard HTML controls (for example, check boxes and radio buttons). When these images appear in a table that contains options, they appear in a separate column.

The Unsatisfied setting specifies which image appears next to options that must be selected, contain required selections, or require input to create a valid configuration. For example: an Option Feature whose Minimum Selections is set to 1 is unsatisfied until the end user selects at least one of its Options. The following nodes can be unsatisfied at runtime: BOM Option Classes, Option Features, Connectors, and Text Features.

For more information, see Section 5.3.1, "Selection State" on page 5-7.

The default Status Indicator images are shown in Figure 20–2 on page 20-10.

Figure 20–2 Default Selection State and Status Indicator Images



Enhanced Check Box Images		Enhanced Radio Button Images		Status Indicator Images	
☐	Not Selected (chkbox_selectable.gif)	○	Not Selected (radiobtn_selectable.gif)	○	Not Selected (selectable_status.gif)
✓	Selected (chkbox_userselected.gif)	●	Selected (radiobtn_userselected.gif)	✓	Selected (userselected_status.gif)
✓	Auto-Selected (chkbox_sysselected.gif)	●	Auto-Selected (radiobtn_sysselected.gif)	✓	Auto-Selected (sysselected_status.gif)
✗	Declined (chkbox_userrejected.gif)	✗	Declined (radiobtn_userrejected.gif)	✗	Declined (userrejected_status.gif)
✗	Auto-Excluded (chkbox_sysexcluded.gif)	✗	Auto-Excluded (radiobtn_sysexcluded.gif)	✗	Auto-Excluded (sysexcluded_status.gif)
▨	▨	▨	▨	●	Unsatisfied (unsatisfied_status.gif)

Note: If you specify different selection state or status indicator images, you must also update the Icon Legend template so that it displays the new images in your UI. See Section 20.3.5.3, "Icon Legend Template" on page 20-22.

20.2.3 Oracle Browser Look and Feel with Step-by-Step Navigation UI Master Template

A UI generated using this template uses the step-by-step navigation method, which means navigation buttons are provided on each UI Page to allow end users to navigate to each page in the UI and access the Configuration Summary page. A progress indicator also appears at the top of each page by default. This indicator shows all of the required steps to create a valid configuration and highlights the current step in the process. You can prevent this image from appearing by deselecting the Show Train for all Multi-page Flows setting, which appears in the template's Pagination and Layout section.

This template uses the Standard Layout to display BOM and non-BOM content. For a description of the Standard Layout and additional information about displaying BOM and non-BOM content, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.2.4 Oracle Browser Look and Feel with Dynamic Tree Navigation UI Master Template

Select this template to display a list of links to each UI Page at runtime. This list reflects the Model's structure and is dynamic, which means Oracle Configurator adds or removes links when an end user adds or deletes component instances at runtime.

This template uses the Standard Layout to display BOM and non-BOM content. For a description of the Standard Layout and additional information about displaying BOM and non-BOM content, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.2.5 Step-by-Step Navigation UI Master Template

Create this template if you want a UI that is similar to one generated using the Oracle Browser Look and Feel with Step-by-Step Navigation UI Master Template, but you want to display BOM and non-BOM content in the Compact Layout.

For a description of what the Compact Layout setting provides and additional information about displaying BOM and non-BOM content, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

For more information about the Oracle Browser Look and Feel with Step-by-Step Navigation UI Master Template, see Section 20.2.3 on page 20-10.

20.2.6 Dynamic Model Tree Navigation UI Master Template

Create this template if you want a UI that is similar to one generated using the Oracle Browser Look and Feel with Dynamic Model Tree Navigation UI Master Template, but you want to display BOM and non-BOM content in the Compact Layout.

For a description of what the Compact Layout setting provides and additional information about displaying BOM and non-BOM content, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

For more information about the Oracle Browser Look and Feel with Dynamic Tree Navigation UI Master Template, see Section 20.2.4 on page 20-11.

20.2.7 Single-Level Side Navigation UI Master Template

A UI that you generate using this template displays a Menu containing a link for each Page in the UI. (When editing a UI, these links appear as elements that are called Page Links.) At runtime, the Menu appears in the top left region of each UI Page. Unlike the Dynamic Model Tree navigation style, the Menu does not reflect the Model's structure.

In the Single-Level Side Navigation Menu, Configurator Developer generates a Page Link only for the first level of nodes beneath the root Model node. In other words, a Page Link appears at runtime for each BOM Option Class and Component that is a child of the Model's root node. After generating the UI using this template, you can edit each Page Link or create new Page Links in the User Interface area of the Workbench.

For more information about Menus and Page Links, see Section 21.5 on page 21-5.

When you use this template, the list of Page Links is static at runtime. That is, Oracle Configurator does not update the list of Page Links when the end user adds or deletes component instances.

This template uses the Compact Layout to display BOM and non-BOM content. For details, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.2.8 Multiple-Level Side Navigation UI Master Template

This template is similar to the Single-Level Side Navigation template, but this template generates a Page Link for the first and second levels of nodes beneath the Model's root node. In other words, it generates Page Links for BOM Option Classes and Components that are children of the root Model node, and for BOM Option Classes and Components that are "nested" one or more levels beneath the root node.

After generating the UI using this template, you can edit each Page Link or create new Page Links in the User Interface area of the Workbench. For more information about Menus and Page Links, see Section 21.5 on page 21-5.

When you use this template, the list of Page Links is static at runtime. That is, Oracle Configurator does not update the list of Page Links when the end user adds or deletes component instances.

For additional details, see Section 20.2.7, "Single-Level Side Navigation UI Master Template" on page 20-12.

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.2.9 Subtab Navigation UI Master Template

A UI generated using this template displays tabs at the top and bottom of each UI Page, and each tab is a link to a UI Page (Page Link). How many tabs are created is based on the Model's structure and the UI Master Template's pagination settings. After generating the UI using this template, you can edit each link or create new links in the User Interface area of the Workbench.

When you use this template, how many links (tabs) appear does not change at runtime. That is, Oracle Configurator does not add or remove links when an end user adds or deletes component instances.

This template uses the Compact Layout to display BOM and non-BOM content. For details, see:

- Section 20.2.2.3, "BOM Content Section" on page 20-7
- Section 20.2.2.4, "Non-BOM Content Section" on page 20-8

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.2.10 Single Page Layout UI Master Template

A UI generated using this template displays all of your Model's options on a single page. You may want to create this kind of UI if your Model is small or if you want your end users to be able to quickly make all required selections without navigating to multiple pages. If your Model contains instantiable Components or Model References, Configurator Developer generates additional UI Pages to display their content.

Because all options appear in a single UI Page, a UI generated using this template does not contain any navigation controls.

Table 20–1 on page 20-3 lists the other default settings for this UI Master Template.

20.3 User Interface Content Templates

When you generate a UI, the UI Content Templates specified by your UI Master Template control the appearance and behavior of the User Interface. These templates contain reusable UI elements such as Layout Regions, Tables, Buttons, Check Boxes, and so on. For a description of all available UI elements, see Chapter 21, "User Interface Structure and Design".

A UI Content Template's type determines the UI element Configurator Developer creates when you generate a UI. For example, Configurator Developer uses:

- Selection Control Templates to display controls that enable an end user to select options during a configuration session
- Button Bar Templates to display buttons that enable an end user to navigate the runtime UI, connect components, save a configuration, and so on
- Message Templates to display the messages that appear at runtime

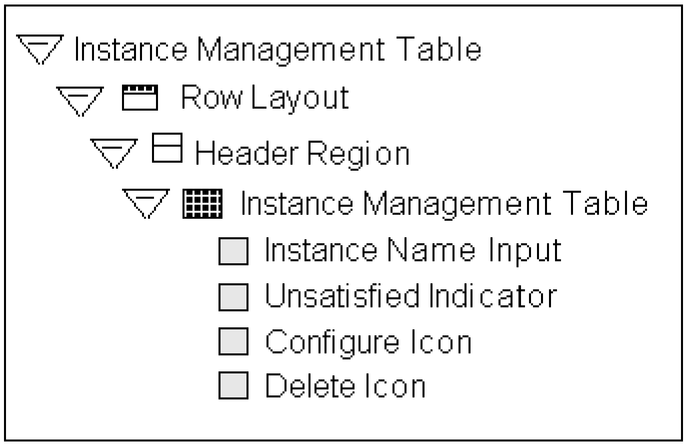
You can also use UI Content Templates to create UI elements after generating a UI. for details, see Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7.

The predefined UI Content Templates appear in the Main area of the Repository in the UI Templates Folder. The various types of Content Templates are grouped within subFolders beneath the UI Templates Folder. The names of these Folders correspond to the type of UI Content Templates they contain. For example, Selection Control, Instance Management, Button Bar, and so on.

The predefined UI Content Templates are read-only and cannot be modified or deleted. However, you can make a copy of any predefined template and then modify it. For details, see Section 31.5, "Editing a User Interface Content Template" on page 31-33.

A UI Content Template is comprised of UI elements. You can view a UI Content Template's structure in the User Interface area of the Workbench. Figure 20–3 on page 20-14 shows the predefined Instance Management Table template.

Figure 20–3 Structure of the Instance Management Table UI Content Template



In this template, Instance Name Input is a Text Input element, while Configure Icon and Delete Icon are both Image elements. For details about these and other UI elements, see Chapter 21, "User Interface Structure and Design".

20.3.1 Specifying How a User Interface Uses Content Templates

The Template Usage setting controls how UI Content Templates are used by a generated UI. You can specify a Template Usage in the following sections of a UI Master Template: BOM Content, Non-BOM Content, Utility Templates, and Message Templates.

Following are the available Template Usages:

- **Incorporate by Reference**: Select this setting if you want the UI to reference the the templates when displaying their content at runtime. In other words, the UI structure will contain a reference to each UI Content Template that uses this setting, and any changes to the template automatically appear at runtime when unit testing the UI from Configurator Developer.

  This is the default setting for the predefined UI Master Templates.

  In the UI structure, elements that are incorporated by reference appear as UI Template References. See Section 21.16.1, "User Interface Template References" on page 21-48.

- **Copy to UI as Page Content**: Select this setting to copy a UI Content Template's structure into the UI. When you select this option, the template is used only to create Page content (UI elements) that can be edited manually in the User Interface area of the Workbench. The advantage to this is that multiple copies of Page content can be made from a single UI Content Template, and you can then modify the content as necessary.

Note: Message, Summary Page, Page Status Area, and Button Bar templates cannot be copied as page content. These templates can only be incorporated by reference and loaded as needed at runtime. Additionally, some predefined templates reference another template (for example, the Multi-Select BOM Item Table with Header template). A nested template is still referenced after you generate a UI, even when you select Copy to UI as Page Content for the parent template.

20.3.1.1 Copying a Template as Page Content

One template might not meet your needs for all Model nodes of a given type (as defined in the Master UI Template). Therefore, you may want to copy certain classes of template content into a UI Page when generating a UI, or convert individual template references into Page content after generating the UI.

For example, when you generate a UI, Configurator Developer uses a single Control Template for all single-selection BOM Option Classes without quantity input. However, depending on how they are used in the Model, you may want some BOM Option Classes to be drop-down lists and some to be radio button groups. Or, you may want to display different sets of Property columns in Item Selection Tables that display different BOM Option Classes. By copying the template into the UI as Page content, you can make these changes by editing the generated UI.

For more information, see:

- Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7
- Section 31.3.2.1, "Converting a UI Template Reference" on page 31-6.

20.3.2 The Predefined Button Bar UI Content Templates

These templates display buttons that appear:

- In various messages (for example, Yes, No, and Cancel)
- In a transaction flow (for example, Cancel, Preview Configuration, and Apply)
- In the Connections Chooser page (for example, Cancel, Set, and Clear)

These templates appear in the predefined Button Bar Templates Folder in the Main area of the Repository.

Note: The predefined Button Bar Templates do not provide a button to perform an intermediate save during a configuration session. Do not add a Save button or link to a template or custom UI unless you know that the host application supports this action.

20.3.2.1 Basic Transaction Button Bar

This template displays the following buttons every UI Page: Cancel, Preview Configuration, and Finish. Recalculate Price and Availability buttons also appear if pricing and ATP are enabled.

20.3.2.2 Connection Chooser Button Bar

This template displays buttons that appear in the Connection Chooser page. They include Cancel, Set Connection, and Clear Connection.

This template is used only by the Connection Chooser Template and is not available as a setting in a UI Master Template. The Connection Chooser Template is described in Section 20.3.3.12 on page 20-19.

20.3.2.3 Preview Page Button Bar

This template displays the buttons that appear in the Configuration Summary page, which include Cancel, Return to Configuration, and Finish.

For more information, see Section 19.5, "The Configuration Summary Page" on page 19-11.

20.3.2.4 Step-by-Step Navigation Bar

This template displays the following buttons on every UI Page: Cancel, Back, Next, and Preview Configuration. Either a Finish or Apply button appears on the last page of a transaction flow, depending on the nesting level. For a definition of nested structure, see Section 20.2.2.2.1 on page 20-5.

If pricing and ATP are enabled, Recalculate Price and Availability buttons also appear.

20.3.2.5 Two-Page Navigation Bar

This template displays the following buttons on both UI Pages in a task flow containing two pages: Cancel, Continue (on the first page), Back (on the second page), and Preview Configuration. Either a Finish or Apply button also appears on the last page of a transaction flow, depending on the nesting level. For a definition of nested structure, see Section 20.2.2.2.1 on page 20-5.

If pricing and ATP are enabled, Recalculate Price and Availability buttons also appear.

20.3.2.6 Yes or No Confirmation Button Bar

This template displays Yes and No buttons that appear in messages created by the various Message Templates. This template is referenced by the Message Templates and is not available as a setting in a UI Master Template.

The predefined Message Templates are described in Section 20.3.4 on page 20-20.

20.3.3 The Predefined Control UI Content Templates

Control templates display UI controls for the various types of selectable nodes in your Model. For example, all of the predefined UI Master Templates use a Multi-Select BOM Item Table for BOM Standard Items that are not mutually exclusive and require a quantity.

These templates appear in the predefined Control Templates Folder in the Main area of the Repository.

The Runtime Display Name setting in the General area of the Workbench determines the default caption for each UI control. For details, see Section 28.9, "Runtime Display Names" on page 28-6.

All of the predefined Control Templates are available as choices when you are defining or customizing a UI Master Template, except where indicated. For details, see Section 20.2.2, "UI Master Template Information and Settings" on page 20-4.

20.3.3.1 BOM Item Table Control Templates

These templates display Required Single Instance Model References, BOM Option Classes, and BOM Standard Items in a table. (For details about the Required Single Instance setting, see on page 29-13.) For example, when an Item Selection Table is associated with a BOM Option Class, the BOM Option Class's children appear as the rows of the table.

Select a template based on whether you want to display quantities, and whether end users can select only one or multiple options.

Following are the available BOM Item Table Templates:

- Single-Select BOM Item Table: This template displays a table containing mutually exclusive BOM Standard Items which users select using radio buttons. The table provides a field for entering a quantity, shows each option's selection state, logic state, description, and pricing and availability (if pricing and ATP are enabled).
- Single-Select BOM Item Table without Quantity: Same as the preceding template, but the table generated by this template does not provide a field for entering a quantity.
- Multi-Select BOM Item Table: Same as the Single-Select BOM Item Table template, but this template is used for BOM Standard Items that are *not* mutually exclusive. (In other words, users select options using check boxes, rather than radio buttons.)
- Multi-Select BOM Item Table without Quantity: Same as the preceding template, but does not provide a field for entering a quantity.

20.3.3.2 BOM Item Status Region Template

This template displays the status, quantity, price and ATP information for a single BOM item.

This template is referenced by the Message Templates and is not available as a setting in a UI Master Template.

20.3.3.3 Instance Management Control Templates

These templates display UI controls for instantiable Components, BOM Model References, and non-BOM Model References and are available when defining BOM and non-BOM content settings. See Section 20.2.2 on page 20-4.

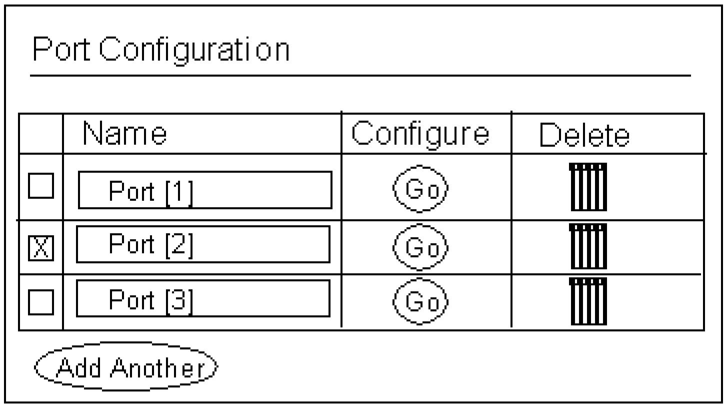
Following are the Instance Management Control Templates:

- The BOM Single Instance Control template displays a check box and button to enable an end user to select and configure an optional single instance of a BOM Model. Therefore, the end user can add only one instance of the BOM Model to the configuration at runtime. (The Optional Single Instance setting is described in Section 29.17.6, "Instances" on page 29-13.)
- The Single Instance Control template displays a check box and a button to enable an end user to select and configure an optional single instance of a Component or Reference to a non-imported Model. (An optional single instance means the end user can add only one instance of the Component or Reference.) This template is the default for instantiable Components and non-imported Model References that are defined as having an optional single instance. See Section 20.2.2.4.1 on page 20-7.
- The BOM Instance Management Table template displays a table with a Quantity field and buttons to configure and add multiple instances of a BOM Model Reference.

- The Instance Management Table template displays a table with buttons to configure and add instances of a Component or non-imported Model References to a non-imported Model that can have multiple instances. The structure of this template is shown in Figure 20–3 on page 20-14.

  An example of how this element might appear at runtime is shown in Figure 20–4 on page 20-18.

Figure 20–4 An Instance Management Table at Runtime



20.3.3.4 Counted Option Table Templates

These templates display a table for Option Features that allow end users to enter quantities for selected options. For details about Counted Option Features, see Section 9.7.1 on page 9-3.

Use the Single-Select Counted Option Table template to display a table that has a Select column with radio buttons, a Quantity column containing input fields, and a Description column that displays read-only text.

The Multi-Select Counted Option Table template displays a table similar to the one described above but displays check boxes, rather than radio buttons, in the Select column.

20.3.3.5 Dynamic and Non-Dynamic Drop-Down Control Templates

These templates display a drop-down list for mutually exclusive BOM Option Classes and Option Features whose Maximum Selections are set to 1. For example, a drop-down list that is associated with an Option Feature node displays all of the Feature’s Options as items in the list.

In the Non-Dynamic Drop-down template, the default Excluded Items Suffix is "[x]" (without the quotes). At runtime, Oracle Configurator inserts a space between the item’s display name and the suffix.

By default, these templates do not show price and availability (ATP) information. To display pricing and ATP within these templates, you can copy one of the predefined templates and customize it to display the pricing and ATP data, or create a new template from scratch. For details, see:

- Section 31.5, "Editing a User Interface Content Template" on page 31-33
- Section 31.4, "Creating a User Interface Content Template" on page 31-30.

20.3.3.6 Enhanced Check Box Group and Enhanced Radio Button Group Control Templates

These templates use image-based versions of check boxes and radio buttons to indicate logic state of options for BOM structure and Option Features. These templates display all options in a group and indicate each option's selection state using the images specified in the Images page of a UI Master Template. For details, see Section 20.2.2.7, "Images Section" on page 20-9.

20.3.3.7 Boolean Feature Check Box Template

This template displays a check box that indicates a Boolean Feature's selection state. You specify the icons used to display status in the Images page of a UI Master Template. See Section 20.2.2.7 on page 20-9.

For more information about Boolean Features, see Section 9.7.4 on page 9-4.

20.3.3.8 Numeric Input Template

This template displays an input field for a Decimal or Integer Feature. At runtime, the input field accepts either a decimal or an integer value, depending on the Feature's type.

20.3.3.9 Text Input Template

This template displays an input field that accepts alphanumeric values for all Text Features in your Model.

This template supports an Initial Value that occupies only one line at runtime. If you need to insert one or more hard returns when specifying a Text Feature's Initial Value in Configurator Developer, create a new UI Content Template for Text Features (for example, by copying the predefined Text Input template), and then open it for editing. Open the Text Input element for editing, and then change the "Display Height (in lines)" setting to the number of rows that the Text Feature's Initial Value requires at runtime (for example 3). Update your UI Master Template to use the new template, and then generate a UI.

20.3.3.10 Read-Only Text Data Template

This template displays a read-only value for Totals and Resources. You can also use this template for read-only presentation of Numeric Features.

20.3.3.11 Connection Control Template

This template displays a Choose Connection button that enables end users to connect components at runtime.

When you generate a new User Interface, Configurator Developer generates a connector control for each Connector in your Model. This control appears in the Connector's parent UI Page and enables an Oracle Configurator user to connect the component to another Model instance at runtime. If a connection is required, an unsatisfied indicator image appears in the control at runtime. For details about indicator images, see Section 20.2.2.7.2 on page 20-10.

For general information about connectivity, see Chapter 8.

20.3.3.12 Connection Chooser Template

This template displays a list of Model instances to which a component can be connected at runtime, and provides buttons that allow an Oracle Configurator end user to set or clear an existing connection. The list appears when an end user clicks a button to create a connection.

20.3.3.13 Connection Navigator Template

This template displays a table containing a list of components that are connected to the associated Component or Model at runtime. Each component appears as a link the end user can click to navigate to that component.

Generating a UI does not create this type of UI element automatically. If you want to display a Connection Navigator Table on a UI Page at runtime, you must create it in the User Interface area of the Workbench. See Section 31.3.4.7, "Creating a Connection Navigator Table" on page 31-11.

20.3.4 The Predefined Message UI Content Templates

These templates display the messages that appear at runtime. For a description of how a UI Master Template uses these templates, see Section 20.2.2.6 on page 20-9.

These templates appear in the predefined Message Templates Folder in the Main area of the Repository.

The following types of messages are available:

- A **Message Box** appears at the top of the current UI Page, above the page content. You may want to display validation warnings and notifications using a Message Box, for example, as it conveys useful information but allows the end user to continue without acknowledging the message.
- A **Modal Message Box** appears at the top of a UI Page, but prevents any actions in the page until the end user acknowledges the message (for example, by clicking Yes or No). The Overridable Contradiction Modal Message Box template displays this type of message.
- When a message appears as a **Dialog Page**, its content fills the entire UI Page. In this case, the end user must acknowledge the message before proceeding.

Following is a description of each predefined Message Template.

20.3.4.1 Notifications Message Box Template

This template displays a Message Box that includes validation messages, pricing and ATP notifications, and any messages generated by Configurator Extensions.

20.3.4.2 Invalid Input Message Box Template

This template displays the message that appears when and end user enters an invalid value at runtime (for example, when a user enters text in a field that requires a numeric value).

20.3.4.3 Overridable Contradiction Message Templates

These templates display messages that appear when a contradiction occurs at runtime. These messages allow the end user to either confirm or cancel the selection that caused the contradiction.

Depending on the type of message you want to display, use either the Overridable Contradiction with Confirmation Dialog Page template or the Overridable Contradiction with Confirmation Modal Message Box template. For a description of each message type, see Section 20.3.4 on page 20-20.

20.3.4.4 Non-Overridable Contradiction Message Box Template

This template displays a message that appears when a contradiction requires the end user's previous action to be cancelled (for example, when the end user makes a selection that causes a resource to be overconsumed).

20.3.4.5 Confirmation Message Templates

These templates display messages that prompt the end user to confirm a requested action, such as exiting a configuration session when the configuration is incomplete.

Confirm Load Instance Dialog Page Template

The Confirm Load Instance Dialog Page template displays a message that appears when an end user is reconfiguring an installed configuration and navigates to a component that was not loaded in the current configuration session. (See Note below.)

Confirm Edit Instance Dialog Page Template

This template displays a message that appears when an end user is reconfiguring an installed configuration and makes a change that requires additional components to be modified in the configuration session. (See Note below.)

Note: For information about reconfiguring installed configurations, see the *Oracle Telecommunications Service Ordering Process Guide*.

Confirm Delete Instance Message Template

This template displays a message that appears when an end user deletes a component from the configuration.

This template is the default for the Confirm Delete Instance setting in a UI Master Template. See Section 20.2.2.6 on page 20-9.

Query Delete Instance Dialog Page Template

This template displays a message that appears when an end user deselects or enters a quantity of 0 (zero) for a BOM Model instance. By default, this message contains Yes and No buttons, and the following text: "You have deselected *instance name*. Do you also want to delete the instance?"

Confirm Save Dialog Page Template

This template displays a message that allows the end user to save an incomplete or invalid configuration.

This template is the default for the Confirm Save/Finish setting in a UI Master Template. See Section 20.2.2.6, "Message Templates Section" on page 20-9.

Confirm Cancel Dialog Page Template

This template displays a message that appears when an end user clicks Cancel to end a configuration session (that is, without saving the configuration).

This template is the default for the Confirm Cancel setting in a UI Master Template. See Section 20.2.2.6 on page 20-9.

20.3.5 Other UI Content Templates

The templates described in this section appear in the predefined Other Templates Folder in the Main area of the Repository.

20.3.5.1 The Combination Status Region Template

Use the Combination Status Region template to determine how status regions display lists of items and status messages at runtime.

This template is not available as a setting in a UI Master Template. You can create a Combination Status Region manually in the User Interface area of the Workbench, but it is optional in a UI.

20.3.5.2 Summary Page Templates

The Summary Page templates display the content of the Configuration Summary page. For details about this page, see Section 19.5, "The Configuration Summary Page" on page 19-11.

Which Summary Page template a UI uses depends on whether the Model contains trackable BOM items and the end user is reconfiguring an installed configuration. BOM Models that contain trackable items support reconfiguration of installed configurations. For more information about reconfiguring installed configurations, see the *Oracle Telecommunications Service Ordering Process Guide*.

If the Model does not contain any trackable BOM items and the end user is not reconfiguring an installed configuration, the UI uses the Summary with Status Region template. This template displays all selected items in a table that includes the following columns: Item, Name and Quantity. If the UI is set up to display pricing and available to promise (ATP) information, Unit List Price, Unit Selling Price, Extended Price, and Availability columns also appear.

If the Model contains trackable BOM items, end users can reconfigure installed configurations of the selected Model. In this case, the UI uses both of the following Summary Templates:

- Upgrade Summary with Status Region, Complete: This template is similar to the standard Summary Page with Status Region template, but also displays a Line Type column by default. This template shows all items in the configuration and displays an icon next to items that have been added, removed, or have changed relative to Oracle Install Base. It also provides a button the end user can click to switch to a summary view generated by the template described below.

  This template is the default for the "Configuration Upgrades Summary Page, Complete" setting in a UI Master Template.

- Upgrade Summary with Status Region, Changes Only: This template shows only the items (and their parents) that have changed relative to Oracle Install Base. It provides a button the end user can click to switch back to the summary view generated by the template described above.

  This template is the default for the "Configuration Upgrades Summary Page, Filtered Only" setting in a UI Master Template.

For details about reconfiguring installed configurations, see the *Oracle Telecommunications Service Ordering Process Guide*.

20.3.5.3 Icon Legend Template

This template displays the default icons that a generated UI displays at runtime to indicate:

- Each option’s selection state (for example, Auto-Selected, Declined, and so on)
- Any options that are unsatisfied

These icons are shown in Figure 20–2 on page 20-10.

This template also displays a UI control that enables end users to hide or show the list and description of the icons.

This template is referenced by other templates, and is therefore not available as a setting in a UI Master Template.

For more information, see:

- Section 5.3.1, "Selection State" on page 5-7
- Section 20.2.2.7.2, "Status Indicator Images" on page 20-10

20.4 Referencing a User Interface Content Template

By default, the predefined UI Master Templates reference UI Content Templates. The benefits of referencing a template, rather than copying it into a UI as content, include:

- The template can be shared by multiple UIs simultaneously
- Changes to a template automatically appear at runtime in any UIs that reference the template

For details, see Section 20.3.1, "Specifying How a User Interface Uses Content Templates" on page 20-14.

The predefined UI Content Templates appear in the Main area of the Repository in the UI Templates Folder. However, you can store UI Content Templates that you create anywhere in the Main area of the Repository.

You select from the available UI Content Templates when creating or modifying a UI Master Template. For example, from the Message Templates page in a UI Master Template, you click Choose to select a template for displaying overridable contradiction messages. When you do this, Configurator Developer displays all UI Content Templates that are stored in the Content Templates Folder in the Main area of the Repository. You then expand sub-Folders to locate and select the template you want to use.

UI Content Templates must be published to make them available to a runtime Oracle Configurator that is launched from a host application (such as Oracle Order Management). All Content Templates that a UI references are published automatically when you publish a Model and UI. See Section 27.2, "Creating a New Model Publication" on page 27-2.

For more information, see:

- Section 31.4.1, "Creating a User Interface Content Template" on page 31-31
- Section 31.5, "Editing a User Interface Content Template" on page 31-33

20.5 Displaying Pricing and ATP Using a UI Content Template

The predefined non-BOM Control Templates do not show pricing and Available to Promise (ATP) information, even if pricing and ATP is enabled. If you want to show pricing and ATP information for non-BOM nodes, you must create your own templates. See Section 31.4, "Creating a User Interface Content Template" on page 31-30.

If you want to show pricing or ATP information for BOM nodes using one of the predefined UI Content Templates, use a template that displays content in a tabular format, rather than one that displays items in a radio button or check box group. In other words, the UI Master Template you select should use the Standard Layout. If your UI Master Template uses the Compact Layout, pricing and ATP information does not appear when you generate the UI. This setting is described in Section 20.2.2.3, "BOM Content Section" on page 20-7.

If you use a custom UI Content Template (that is, one that you created), you can display prices and ATP information using other types of UI controls, such as radio buttons or other types of layout.

If pricing or ATP is not enabled, any related columns or buttons that are provided by the predefined Content Templates do not appear at runtime. If you are using a custom UI Content Template, it is recommended that you define a display condition to hide or disable pricing and ATP columns and UI controls whenever pricing or ATP is disabled.

To do this, the define a condition and specify Configuration Session as the object, and then select one of the following session properties:

- PricingEnabled
- ListPriceEnabled
- SellingPriceEnabled
- ATPEnabled
- PriceAndATPDisabled

For more information, see:

- Section 19.6, "Displaying Prices and Available to Promise Dates" on page 19-12
- Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33

21 User Interface Structure and Design

This chapter describes the structure of a generated User Interface and the various types of elements that are available when editing a UI.

This chapter includes the following sections:

User Interface design directly affects how a configuration model performs in a runtime Oracle Configurator. For information about how to construct a UI for maximum runtime performance, see the *Oracle Configurator Modeling Guide*.

21.1 Introduction

There are two ways to create a User Interface in Configurator Developer. You can generate either:

- A UI that is based on your Model's structure
- An "empty" UI (that is, one that is not based on the Model's structure and for which you define all content)

When you generate a UI that is based on your Model's structure, Configurator Developer uses the UI Master Template you specify to generate the UI's content, including the option selection controls for each Model node, navigation controls, UI Pages, and so on. For more information, see Section 20.2, "User Interface Master Templates" on page 20-2.

Generate an empty UI if you want to build a UI from scratch and do not want it to be based on the Model's structure. When you generate an empty UI, Configurator Developer creates a single empty UI Page in the primary navigation method specified by the UI Master Template, and the UI's basic structure. This structure includes the UI Definition node and the Pages, Menus, and Page Flows Folders (see Figure 21–1 on page 21-3). These Folders contain all UI elements you create when building the UI. For example, you create Menus in the Menus Folder, UI Pages in the Pages Folder, and so on. (You create individual UI elements, such as option selection controls, buttons, images, and so on, within UI Pages.) These tasks are explained in Section 31.3, "Editing a User Interface" on page 31-2.

User Interfaces that you generate in Configurator Developer use the Oracle Applications Framework and conform to Oracle's standards for browser accessibility, look, and feel. They also assure a seamless integration with other Oracle Applications products.

21.1.1 Unit Testing

You do not have to generate a UI to unit test a configuration model in Configurator Developer. If you have not yet generated a UI, you can unit test it using the Model Debugger. For more information, see Section 22.2, "The Model Debugger" on page 22-2.

You can unit test a configuration model in a generated UI at any time. For details, see Section 22.3, "Unit Testing a Generated User Interface" on page 22-3.

21.1.2 User Interfaces and the Runtime Oracle Configurator

For a configuration model to be available in a host application such as Oracle Order Management, you must generate a UI in Configurator Developer and then publish both the Model and the UI. Publishing is explained in Chapter 23.

A host application launches the Generic Configurator User Interface when a user makes a request to configure an item that is based on a BOM Model that was never published from Configurator Developer, or when no UI is found that matches the provided applicability parameters. For more information about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

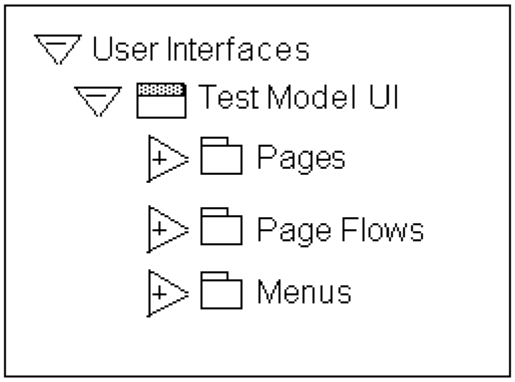
21.1.3 Custom User Interfaces

You can optionally use a different application to create a custom user interface that accesses the Configuration Interface Object (CIO). This implementation is not described directly in any Oracle Configurator documentation. For details about the CIO, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

21.2 User Interface Structure

A UI has a hierarchical structure. When you create an empty UI, or generate a UI that is based on the Model's structure, you can view or modify its structure in the User Interface area of the Workbench. See Figure 21–1 on page 21-3.

Figure 21–1 Collapsed User Interface Structure and System Folders in Configurator Developer



The root node of a UI is called the **UI Definition**. You can open this node for editing to modify settings and review general information about the UI. For details, see Section 21.3, "User Interface Definition" on page 21-3.

A UI's structure also contains the following predefined Folders, which contain automatically generated and manually created UI content:

- **Pages**: This Folder contains all UI Pages.

  For details, see Section 21.4, "User Interface Pages" on page 21-4 and Section 21.8, "Layout Regions" on page 21-8.

- **Page Flows**: This Folder contains one or more Page Flows.

  For details, see Section 21.6, "Page Flows and Page References" on page 21-7.

- **Menus**: This Folder contains one or more Menus.

  For details, see Section 21.5, "Menus and Page Links" on page 21-5.

21.3 User Interface Definition

The UI Definition is the root node of a User Interface's structure. In Figure 21–1 on page 21-3, the UI Definition node is called Test Model UI.

The UI Definition node's details page contains general information about the UI, including:

- The UI Master Template used to create the UI
- The UI's primary navigation style
- Whether pricing and ATP information is displayed and how it is updated and recalculated at runtime
- Whether the UI can be refreshed
- The initial Page Flow, Menu, or UI Page that appears at runtime
- All Message Templates used by the UI
- All standard or control and indicator images used in the UI (such as Enhanced Check Boxes, Enhanced Radio Buttons, and Indicator Images)
- Any referenced UIs

The UI Master Template used to generate the UI determines the default values for all settings and information that appears in the UI Definition's details page.

You can perform the following operations on the UI Definition node:

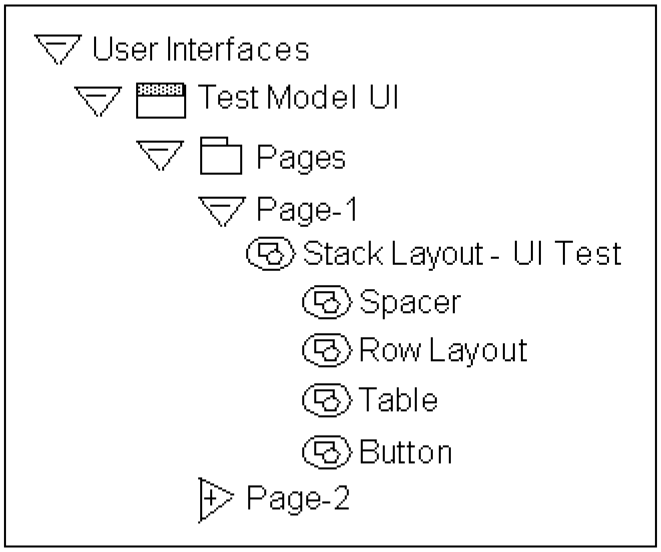
- Refresh: Updates the UI with any recent changes made to the Model structure.
- Delete: Deletes the selected UI.
- Edit: You can change the selected UI Content Templates and images specified by the UI Master Template, select different referenced UIs, modify Pricing and ATP settings, and update the UI name or description.

  For details, see Section 31.3.1, "Modifying the User Interface Definition" on page 31-3.

21.4 User Interface Pages

In the User Interface area of the Workbench, UI Pages appear in the Pages Folder and represent a page that an end user can navigate to in the runtime User Interface. In the UI structure, UI Pages appear as top-level Layout Region UI elements. These elements contain substructure that represents the page's content, which may include item selection controls, text, images, buttons, and so on. See Figure 21–2 on page 21-4.

Figure 21–2 User Interface Page Structure and Content



When you generate a UI that is based on the Model's structure, Configurator Developer generates UI Pages based on the Pagination settings specified in your UI Master Template. These settings are described in Section 20.2.2.2, "Pagination and Layout Section" on page 20-5. For example, when you generate a UI Configurator Developer creates a UI Page for each Model, BOM Model, Component, and BOM Option Class node in your Model (assuming the Display in User Interface setting is selected for each node).

When you create a new UI Page, you must select an Associated Model Node. See Section 21.14, "User Interface Elements and Associated Model Nodes" on page 21-43. After selecting a Model node and saving the UI Page, you cannot change the Page's associated Model node.

You can also modify existing UI Pages by adding UI content. The types of UI elements you can create include Buttons, Images, selection controls, and Layout Regions. All elements that appear as children of a UI Page element in the UI structure appear on the same page at runtime.

See Section 31.3.3, "Creating a User Interface Page" on page 31-6.

21.4.1 Empty User Interface Pages

When you generate a UI and your UI Master Template creates a UI Page for a node that has no child nodes, the UI Page has no content. Empty UI Pages appear in the UI's structure so you can add content to them when customizing the UI. However, if the UI has a Menu, Configurator Developer does not automatically create Page Links to any empty UI Pages. If the UI contains a Page Flow instead of a Menu, Configurator Developer does not add empty UI Pages to the Page Flow. Menus and Page Links are described in Section 21.5 on page 21-5.

If you add child structure to a node that previously generated an empty UI Page, or add content to an empty UI Page, you must refresh the UI. Depending on the UI's primary navigation method, refreshing the UI either adds the previously empty UI Page to the Page Flow, or creates a new Page Link within the Menu.

If the UI contains any empty UI Pages when a configuration session begins, the runtime Oracle Configurator ignores them. For example, a UI that uses the Step-by-Step navigation style skips any empty pages when the end user navigates from one page to the next. This ensures that Oracle Configurator end users do not see empty pages during a configuration session.

If *all* of the UI Pages in a UI are empty when a configuration session begins, Oracle Configurator displays the Configuration Summary page.

For more information, see:

- Section 21.5, "Menus and Page Links" on page 21-5
- Section 21.6, "Page Flows and Page References" on page 21-7
- Section 19.2, "Refreshing a User Interface" on page 19-2
- Section 19.5, "The Configuration Summary Page" on page 19-11

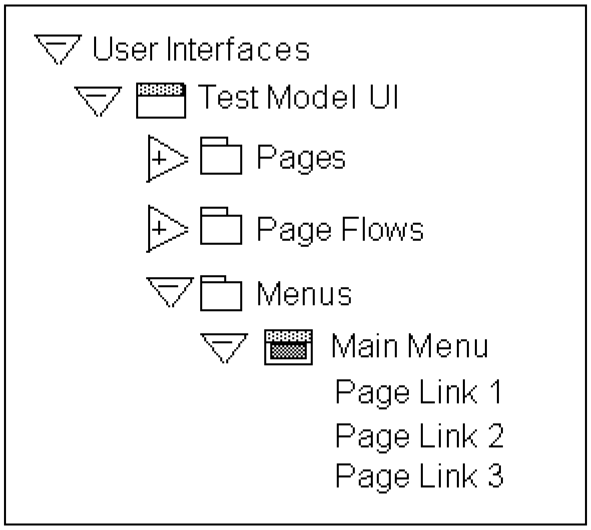
21.5 Menus and Page Links

A Menu is a group of links that enable end users to navigate the runtime User Interface. These links are called Page Links, and the target of each link is always a UI Page.

When viewing the UI structure in Configurator Developer, each Page Link appears as a child of the Menu. See Figure 21–3 on page 21-6.

A Menu is linked to the UI Definition via the Primary Menu setting, and serves as an Oracle Configurator end user's "point of entry" into the UI. (The UI Definition is described in Section 21.3 on page 21-3.)

Figure 21–3 User Interface Structure in Configurator Developer with Menu and Page Links



When you generate a UI, the UI Master Template's Primary Navigation setting controls whether Configurator Developer generates a Menu, and if it does, the Menu's type.

Following are the available Menu types:

- Single Level Side Menu (see Section 20.2.7 on page 20-12)
- Multi-Level Side Menu (see Section 20.2.8 on page 20-12)
- Model Tree Side Menu (see Section 20.2.6 on page 20-11)
- Subtab Layout (see Section 20.2.9 on page 20-12)

Additionally, when you generate:

- A UI that is based on the Model structure, Configurator Developer generates Page Links to each UI Page.
- An empty UI, the Menu contains a single Page Link that points to the only UI Page in the UI (this page is empty by default).

Depending on the Menu's type, its Page Links appear in either the top left region of each UI Page, or as sub-tabs at the top and bottom of each Page at runtime. Figure 21–4 on page 21-7 shows a Menu that appears at the top left of each UI Page.

Configurator Developer does not generate a Menu if the UI Master Template's primary navigation style is Step-by-Step or Single Page. In the Step-by-Step case, Configurator Developer generates a Page Flow instead of a Menu. For details, see Section 21.6, "Page Flows and Page References" on page 21-7. If the navigation style is Single Page, the UI does not require any navigation controls, so none are created.

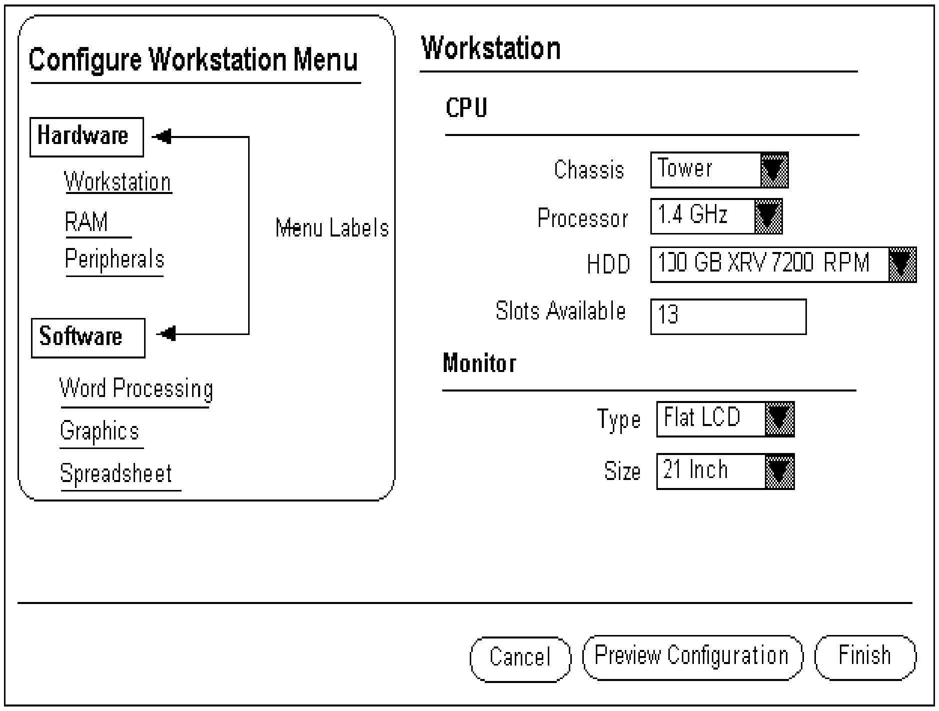
When a UI's primary navigation style is Dynamic Tree, the UI contains a Model Tree Side Menu. This type of Menu does not contain any Page Links when viewed in Configurator Developer. This is because a Model Tree Side Menu displays links to Pages in a hierarchical arrangement that is based on the Model structure and cannot be modified in Configurator Developer.

See Section 31.3.5, "Creating a Menu" on page 31-25.

21.5.1 Menu Labels

Create a Menu Label to group a set of Page Links within a Multi-Level Side Menu. For example, your Model contains several parts and the UI has several pages for configuring each part. You can use Menu Labels to separate the Page Links for each part of the product into groups. See Figure 21–4 on page 21-7.

Figure 21–4 Side Menu and Menu Labels at Runtime



Configure Workstation Menu

- Hardware
  - Workstation
  - RAM
  - Peripherals
- Software
  - Word Processing
  - Graphics
  - Spreadsheet

Workstation

CPU

Chassis	Tower
Processor	1.4 GHz
HDD	100 GB XRV 7200 RPM
Slots Available	13

Monitor

Type	Flat LCD
Size	21 Inch

See Section 31.3.6, "Creating a Menu Label" on page 31-25.

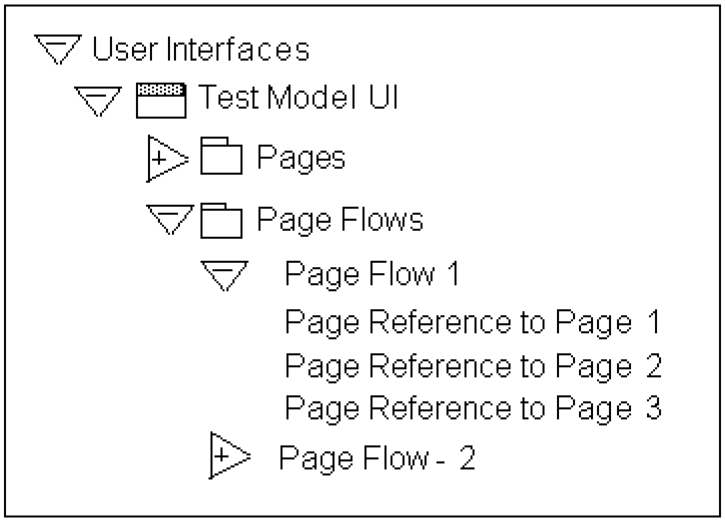
21.6 Page Flows and Page References

A Page Flow defines a set of Pages that an Oracle Configurator end user accesses sequentially using navigation buttons, such as Back and Next. Configurator Developer automatically generates a Page Flow when your UI Master Template's primary navigation style is Step-by-Step.

In the UI structure, Page References appear as children of a Page Flow, as shown in Figure 21–5 on page 21-8. Each Page Reference has a UI Page as its target. The order in which Pages are accessed at runtime is defined by the order of the Page References in the UI structure.

In other words, the Page that is the target of the first Page Reference in a Page Flow (the one at the top in the hierarchy) appears first at runtime, the Page below it appears next, and so on. Reordering Page References in the UI structure changes the order in which Pages appear in a Step-by-Step UI.

Figure 21–5 Page Flows and Page References in the UI Structure



After generating an empty UI, you can create a Page Flow in the Page Flows Folder, and then create Page References (links) to each UI Page. When creating a Page Reference, you select a UI Page as its target. When creating a Page Reference, Pages in a referenced UI are available as targets only if they are within the scope of the Page Flow's page base. For more information about page scope and a definition of page base, see Section 21.15, "Associated Model Nodes and Page Scope" on page 21-44.

For more information, see:

- Section 21.7, "Limitations when Creating UI Pages, Page Flows, and Menus" on page 21-8
- Section 31.3.8, "Creating a Page Flow" on page 31-26

21.7 Limitations when Creating UI Pages, Page Flows, and Menus

A UI may have one or more Page Flows or Menus. When a UI contains multiple Page Flows or Menus, each one must refer to (and contain) different UI Pages. Therefore, when creating a Page Link, you cannot select a Page that already exists within another Page Flow as the link's target. Similarly, you cannot create a Page Reference to a Page that is already linked to a Menu.

However, you can enable end users to navigate from a UI Page that is linked to a Menu, to a UI Page that is part of a Page Flow. To do this, create a Button or Image element and assign it to the "Go to Page" action. For details, see Section 21.13, "User Interface Actions" on page 21-38.

21.8 Layout Regions

A Layout Region contains and organizes UI Page content, which includes other types of UI elements such as a Radio Button Group, Images, or an Item Selection Table. This type of element is not visible in the runtime UI.

When viewing the UI's structure, all UI elements that appear as children of a Layout Region are arranged and displayed according to the region's type at runtime. For example, if you create several Button UI elements under a Stack Layout Region, the buttons are "stacked" at runtime. In other words, they are displayed above one another. See Figure 21–6 on page 21-11. Similarly, to display UI elements in a table, create a Table Layout element and then create the elements you want to appear in the table under the Table Layout.

Tip: To understand how Layout Regions are used, generate a UI that is based on the Model structure using one of the predefined UI Master Templates. Then, review the structure of specific UI Pages carefully in the User Interface area of the Workbench, noting the relationship between Layout Regions and their child UI elements in the structure. Finally, unit test the UI and review each UI Page to see how the elements are arranged at runtime.

When viewing the UI's structure in Configurator Developer, elements that exist at the same level (for example, within the same parent) appear one above the other. The element that appears first in the UI's structure appears first within its parent element at runtime. The element below it in the structure appears next at runtime, and so on, from left to right. For an example, see Figure 21–7, "Runtime Display of UI Elements Within a Row Layout" on page 21-12.

A Layout Region is usually associated with a Component, BOM Model, or BOM Option Class. However, you can, for example, create a Layout Region to represent a single Option Feature with all of its Options displayed as individual controls (such as images with Select actions).

Note: For more information about the UI elements described in this chapter and examples of how they can be used, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

Following are the available Layout Region types:

- Stack Layout on page 21-11
- Table Layout on page 21-11
- Row Layout on page 21-12
- Flow Layout on page 21-13
- Cell Format on page 21-14
- Header Region on page 21-14
- HideShow Region on page 21-16
- Switcher and Case Regions on page 21-27
- Bulleted List on page 21-16

21.8.1 Layout Region Variations

You can create a Basic Layout Region or a Node List Layout Region.

After indicating whether you want to create a Basic or Node List Layout Region, you specify its type. A Layout Region's type controls how its associated UI content is displayed. Layout Region Types include Row Layout, Flow Layout, Table Layout, Header Region, and so on.

To create a Layout Region, see Section 31.3.4.4 on page 31-8.

21.8.1.1 Basic Layout Region

A Basic Layout Region contains only what you specify in Configurator Developer when you create it and, unlike a List Layout Region, it is not populated with additional content at runtime.

21.8.1.2 List Layout Regions

There are two types of List Layout Regions: Node List Layout Regions and Message List Layout Regions.

A **Node List Layout Region** repeats its contents once for each of its associated Model node's children.

You can associate a Node List Layout Region with the following Model structure nodes:

- BOM Model
- BOM Option Class
- Option Feature

For example, you want check boxes to appear next to each of a Feature's Options at runtime. In the User Interface area of the Workbench, you create a Node List Layout Region and associate it with the Feature. In the Node List Layout Region, you then create a single Row Layout Region containing a Check Box and a Styled Text element (to display the state and label for each of the Feature's Options). At runtime, a check box and text is generated and appears in a separate Row Layout for each of the Feature's Options. (An example of a Row Layout is shown in Figure 21–7 on page 21-12.)

You cannot create a Node List Layout Region under a Message List Layout Region.

A **Message List Layout Region** repeats its contents once for each message of the specified type.

Following are the available Message Types:

- Invalid Items
- Unsatisfied Items
- Pricing Notifications
- ATP Notifications
- Configurator Extension Messages
- Contradiction Reasons
- Contradiction Consequences

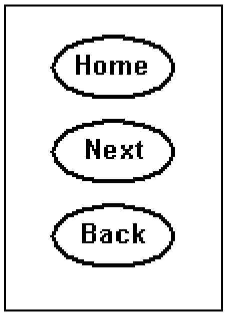
For example, you want to add content to a Notifications Message Box. You create a Message List Layout Region using a Stack Layout, and select a Message Type of Pricing Notifications. At runtime, the contents of the Message List Layout Region are rendered once for each message, and the messages are displayed in a vertical list (that is, in a Stack Layout).

Message List Layout Regions are not associated with Model structure nodes. When creating a Message List Layout Region, you specify a Message Type, and optionally a display or enabled condition. The Message Type determines the class of message that is displayed in the region at runtime. For example, if the Message Type is Invalid Items, the Message List Layout Region lists each invalid item in the configuration. If no message of the specified type exists, then the list is empty.

21.8.2 Stack Layout

Create a Stack Layout when you want to arrange UI content vertically. For example, you want to display three buttons, one above the other, within the same UI Page. You create a Stack Layout, and then create three Button elements under the Stack Layout. Figure 21–6 on page 21-11 shows how the buttons might appear at runtime.

Figure 21–6 Buttons Displayed within a Stack Layout



You can associate a Stack Layout with any type of Model node.

To create a Stack Layout, see Section 31.3.4.4 on page 31-8.

21.8.3 Table Layout

A Table Layout element arranges its child elements in cells like a table at runtime, but it does not display any gridlines. This element contains a Row Layout for each row, with each cell specified by the contents of the Row Layouts.

When you create a Table Layout element, you must always create a Row Layout and Cell Format elements to complete the structure. (These elements are described in Section 21.8.4 on page 21-12 and Section 21.8.6 on page 21-14, respectively.) In other words, each Table Layout must have at least one child Row Layout, and each Row Layout must have a child Cell Format. Then, add content to the Cell Format region. Do not skip regions in this structure as this can result in incorrect HTML output.

For example, you want to create a table for BOM Option Class X, which contains three BOM Standard Items. In the User Interface area of the Workbench:

- Create a Node List Table Layout, and associate it with BOM Option Class X.
- Create a Row Layout within (as a child of) the Table Layout.
- Create a Cell Format within the Row Layout.
- Create a Check Box, Text Input, and Styled Text element within the Cell Format, and enter a display name (caption) of Select, Quantity, and Name, respectively.

  When viewing the UI’s structure in the User Interface area of the Workbench, these elements appear as the Row Layout’s children.

At runtime, the three BOM Standard Items appear in a table similar to the one shown in Figure 31–1 on page 31-10.

A Table Layout is useful when other Layout Regions do not provide enough flexibility for arranging a particular set of content. For example, instead of using a standard list or a table control, you may want to present a set of Feature Options as an arrangement of images and text with associated actions. You can also nest Table Layouts for even greater control over page appearance.

A Table Layout is different from an Item Selection Table or Summary Table. In Item Selection and Summary Tables, you specify the cell contents individually and the cells can be unrelated, rather than being determined by the intersection of row and column specifications. Also, Item Selection and Summary tables display with visible borders and gridlines, whereas a Table Layout generally does not.

When you generate a UI, Configurator Developer controls the alignment of similar Row Layout-based content by wrapping the content in Table Layouts.

Typically, you want a Table Layout to fill the entire space in which it is rendered. Therefore, a Table Layout's default Width is 100%. If you specify a smaller percentage, the starting or ending alignment of its content might not appear as you expect it to (since this type of region does not automatically expand to fill the available space). However, you can specify a number of pixels (300), or a smaller percentage if necessary.

You can associate a Table Layout with any type of Model node.

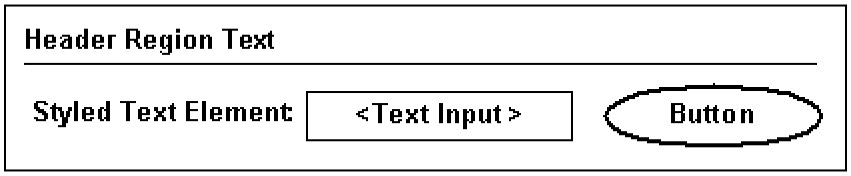
To create a Table Layout, see Section 31.3.4.4 on page 31-8.

21.8.4 Row Layout

Use a Row Layout to arrange content horizontally in a row and align content both horizontally and vertically. This type of element can exist alone or be used to define a row within a Table Layout Region (see "Table Layout" on page 21-11).

For example, you create a Row Layout under a Header Region, and then create the following elements beneath the Row Layout: Styled Text, Text Input, and Button. At runtime, the Styled Text, Text Input, and Button elements appear in a row, from left to right, as shown in Figure 21–7 on page 21-12.

Figure 21–7 Runtime Display of UI Elements Within a Row Layout



Horizontal Alignment settings a Row Layout include Start, Center, and End. Vertical Alignment settings include Top, Middle, and Bottom. You can also specify the element's total width in either pixels, or as a percentage of the Page.

You can associate a Row Layout with any type of Model node.

When editing a UI, you can create a Row Layout as a child of the following UI elements:

- Row Layout
- Table Layout
- Stack Layout
- Flow Layout
- Cell Format
- Connection Targets Table

- Content Container
- Header Region
- Instance Management Table
- Instance Management Control
- Item Selection Table
- Summary Table
- Case Region

To create a Row Layout, see Section 31.3.4.4 on page 31-8.

Note: For important information about defining a display condition for an element that is a child of a Row Layout, see Section 21.11 on page 21-33.

21.8.5 Flow Layout

Like a Row Layout, a Flow Layout also arranges UI content horizontally. (See Section 21.8.4, "Row Layout" on page 21-12.) However, a Flow Layout displays content from either left to right or right to left (depending on the Web browser's localization settings) and automatically wraps content when necessary. You may want to use a Flow Layout instead of a Row Layout when precise vertical alignment of the element's content is not essential.

You can associate a Flow Layout with any type of Model node.

When editing a UI, you can create a Flow Layout as a child of the following UI elements:

- Row Layout
- Table Layout
- Stack Layout
- Flow Layout
- Cell Format
- Connection Targets Table
- Content Container
- Header Region
- Case Region
- Instance Management Table
- Instance Management Control
- Item Selection Table
- Summary Table

To create a Flow Layout, see Section 31.3.4.4 on page 31-8.

21.8.6 Cell Format

Use Cell Format elements to create and format cells within a Row Layout region. (See Section 21.8.4, "Row Layout" on page 21-12.) A Cell Format also allows you to control the width and horizontal and vertical alignment of its content.

You can associate a Cell Format with any type of Model node.

If a display condition prevents a Cell Format element from appearing at runtime, it appears as an empty cell in the row in which it is defined. Defining a display condition is explained in Section 21.11 on page 21-33.

If you need to control the alignment of the Cell Format's content, use the Vertical Alignment and Horizontal Alignment settings. For example, if you want text that appears within the Cell Format to appear at the top of the region, set the Vertical Alignment to Top and the Horizontal Alignment to Start.

When editing a UI, you can create a Cell Format as a child of the following UI elements:

- Case Region
- Cell Format
- Content Container
- Flow Layout
- Header Region
- Row Layout
- Stack Layout
- Table Layout
- Instance Management Control

To create a Cell Format element, see Section 31.3.4.9 on page 31-13.

Note: For important information about defining a display condition for an element that is a child of a Row Layout, see Section 21.11 on page 21-33.

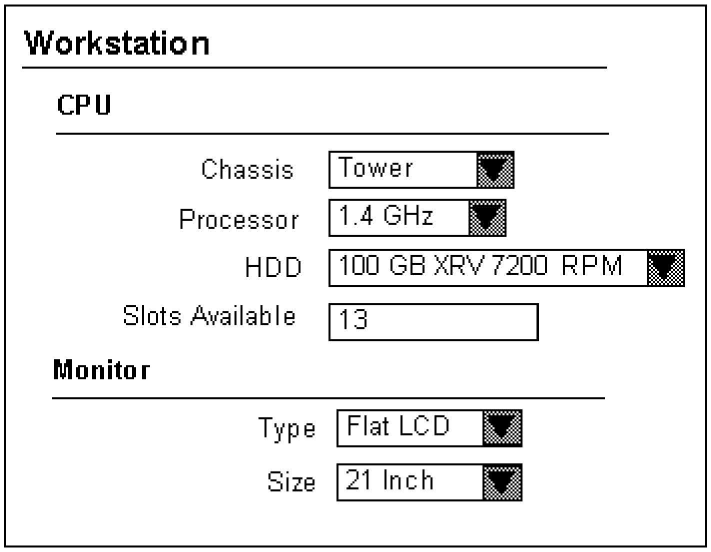
21.8.7 Header Region

A Header Region represents a labeled subdivision of a page and displays its contents beneath a title and a horizontal rule. A Header Region's content is stacked vertically under the title area, making it a good choice for laying out Model content.

Headers can be nested into Subheaders and Sub-subheaders, causing the label style and indentation to be automatically adjusted. Configurator Developer does not limit how many levels of sub-headers you can create.

Figure 21–8 on page 21-15 shows how a Header Region that contains two sub-Header Regions appears at runtime. In this example, the Header Region is called Workstation, and the sub-Header Regions are called CPU and Monitor.

Figure 21–8 Header Region and Two Nested Header Regions at Runtime


Workstation

CPU

Chassis: Tower

Processor: 1.4 GHz

HDD: 100 GB XRV 7200 RPM

Slots Available: 13

Monitor

Type: Flat LCD

Size: 21 Inch

When viewing the UI's structure in Configurator Developer, the selection controls shown in Figure 21–8 (such as Chassis) appear as Row Layout elements, and are children of a Header Region element (that is, CPU and Monitor). See Section 21.8.4, "Row Layout" on page 21-12.

When generating a UI, Configurator Developer may use Header (and subheader and sub-subheader) regions to group the contents of required single-instance Components and BOM Option Classes. Additionally, if the pagination settings in your UI Master Template for nested Components and BOM Option Classes are set to Add to Parent Page, each child element displays with its own header. This setting is explained in Section 20.2.2.2.1, "Defining Custom Pagination and Layout" on page 20-5.

You can associate a Header Region with any type of Model node.

When editing a UI, you can create a Header Region as a child of the following UI elements:

- Case Region
- Cell Format
- Content Container
- Flow Layout
- Header Region
- Row Layout
- Stack Layout
- Table Layout
- Instance Management Control

To create a Header Region, see Section 31.3.4.4 on page 31-8.

21.8.8 HideShow Region

Use this element to enable an end user to selectively hide or display UI content. At runtime, a toggle-style UI control enables the end user to collapse or expand the region, thereby displaying or hiding the region's contents.

When viewing the UI's structure in Configurator Developer, all elements that are children of a HideShow Region represent content that an end user can choose to hide or display.

When editing a UI, you can create a HideShow Region as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Flow Layout
- Instance Management Control
- Instance Management Table
- Connection Targets Table

To create a HideShow Region, see Section 31.3.4.31 on page 31-24.

21.8.9 Bulleted List

This element displays a bullet character before each of its children at runtime. All of a Bulleted List's child elements must be Text elements (for example, Styled Text or Input Text). You can associate a Bulleted List with any type of Model node.

When editing a UI, you can create a Bulleted List as a child of the following UI elements:

- Cell Format
- Content Container
- Connections Target Table
- Flow Layout
- Header Region
- Row Layout
- Stack Layout
- Table Layout
- Case Region

- Instance Management Control

To create a Bulleted List, see Section 31.3.4.4 on page 31-8.

21.9 Basic User Interface Elements

The elements described in this section appear in the "Basic" category when you are creating a UI element in the User Interface area of the Workbench.

For a description of elements that appear in the "Other" category, see Section 21.10, "Other User Interface Elements" on page 21-27.

21.9.1 Styled Text

Create this element to display text in a UI. For example, you may want to provide additional information about a specific option to help end users select the one that meets their requirements.

When defining a Styled Text element, you can either enter the text that you want to display directly, or choose to derive it from one of its associated node's System Properties, User Properties, or a Configuration Session Property.

To display text as a hypertext link at runtime, create a Text Link. See Section 21.9.4, "Text Link" on page 21-19.

When editing a UI, you can:

- Associate a Styled Text element with any type of Model node
- Create a Styled Text element as a child of any other UI element.

When editing a UI, you can create a Styled Text element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

To create a Styled Text element, see Section 31.3.4.10 on page 31-14.

21.9.2 Static Styled Text

This UI element is similar to a Styled Text element, but its content is specified as a simple text string. That is, it cannot derive its content from node or session properties. You can use a Static Styled Text element anywhere you need to display some hard-coded text (text that is not Model or session-related).

For example, this element is used in the predefined Message Templates to display boilerplate text such as "Select Yes to continue or No to return to the configuration."

To display text as a hypertext link at runtime, create a Text Link. See Section 21.9.4, "Text Link" on page 21-19.

When editing a UI, you can create an Enhanced Check Box element as a child of the same UI elements as those listed in Section 21.9.1, "Styled Text" on page 21-17.

To create a Static Styled Text element, see Section 31.3.4.10 on page 31-14.

21.9.3 Formatted Text

This UI element is similar to the Raw Text element, but it supports only specific HTML text formatting tags along with the text to be displayed at runtime. In other words, it accepts a limited set of HTML markup and displays formatted results in the runtime UI.

This element supports the following HTML markup:

- <br>
- <hr>
- <li>
- <pre>
- <ol>
- <ul>
- <small>
- <p>
- <b>
- <i>
- <tt>
- <span>
- <big>
- <a>

The Formatted Text element also supports the following HTML entities:

- &lt;
- &gt;
- &amp;
- &reg;
- &copy;
- &nbsp;

- &quot

When editing a UI, you can create a Formatted Text element as a child of the same UI elements as those listed in Section 21.9.1, "Styled Text" on page 21-17.

To create a Formatted Text element, see Section 31.3.4.12 on page 31-15.

21.9.4 Text Link

Create this element when you want to display a hypertext link in the UI. When defining a Text Link, you can either enter the text that appears at runtime directly, or choose to derive it from one of the element's associated node's System Properties, User Properties, or a Configuration Session Property. This is described in Section 21.12, "User Interface Element Captions and Details" on page 21-35.

For a list of the available actions you can assign to this element, see Section 21.13, "User Interface Actions" on page 21-38.

To create a Text Link, see Section 31.3.4.13 on page 31-15.

21.9.5 Image

Use this element to use an image to display a Model node at runtime. For example, you may want to display a picture of each item in BOM Option Class. To do this, you create an Image element, select the BOM Option Class as its associated Model node, and use the Image Source setting to indicate the image file to use.

To enable end users to execute an action by clicking an image, create an Image Button. See, Section 21.9.6, "Image Button" on page 21-20.

For details about where an image file must be located to appear at runtime, see Section 20.2.2.7 on page 20-9.

When editing a UI, you can:

- Associate an Image with any type of Model node
- Create an Image element as a child of any other UI element.

When editing a UI, you can create an Image element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control

- Connection Targets Table

To create an Image element, see Section 31.3.4.15 on page 31-16.

21.9.6 Image Button

Create this element to display an image with an associated action, such as Go to Page, Add Instance, or Open URL. In other words, an Oracle Configurator end user can click the image you specify to navigate to another page in the UI or to an external Web page, for example.

For a list of the available actions you can assign to this element, see Section 21.13, "User Interface Actions" on page 21-38.

To create an Image Button, see Section 31.3.4.16 on page 31-17.

Example 21–1 Using an Image Button at Runtime

You want to represent selectable items as images at runtime. Doing this provides the following runtime behavior:

- Clicking on the image selects the option if it is not selected, or deselects it if it is selected
- The image is different depending on whether the option is selected, not selected, or excluded

To do this, perform the following:

- Create a Switcher element and specify a Switcher condition of Selection State.

  For details, see Section 31.3.4.30 on page 31-23.
- Create three Case elements within the Switcher element and specify the possible states of the option (selected, not selected, excluded) as the Case Value for the Switcher to display one of the images.
- Create three separate Image Button elements inside each Case element, each with a different image URL.
- Assign a Select action to the Images for the "not selected" and "excluded" cases. (The option in this case is the Image’s associated Model node.)

  For details about UI actions, see Section 21.13 on page 21-38.
- Assign a Deselect action to the Image Button for the "selected" case.

When editing a UI, you can create an Image Button element as a child of the same UI elements as any Layout Region. See Section 21.8, "Layout Regions" on page 21-8.

21.9.7 Standard Button

Create this element to define commonly used buttons, like those used by the predefined Button Bar UI Content Templates. (For example, Apply, Finish, Yes, No, Back, and Next.)

When you create a Standard Button, the Button Type you select determines the Button’s runtime action, UI caption, and access keys. For a list of actions that can be associated with a Standard Button and the access keys for each, see Table 21–1 on page 21-37.

When editing a UI, you can create a Standard Button element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

When editing a UI, you can create a Standard Button element as a child of any type of Layout Region. See Section 21.8, "Layout Regions" on page 21-8.

To create a Standard Button, see Section 31.3.4.17 on page 31-17.

21.9.8 Custom Button

Create a Custom Button when you want to:

- Assign an action that a Standard Button does not provide (see Section 21.9.7, "Standard Button" on page 21-20)

  For example, you can create a button to add a Component to the configuration, navigate to a specific page, or update the list prices for all selected items. You can also perform custom actions by associating a button with a Command Event to trigger a Configurator Extension. For more information, see Section 21.13, "User Interface Actions" on page 21-38.

- Specify how the button's UI caption is created at runtime

  See Section 21.12, "User Interface Element Captions and Details" on page 21-35.

- Define the button's rollover text (that is, the text that appears when the end user places the mouse over the Button, or navigates to the Button by pressing the Tab key)

When editing a UI, you can create a Custom Button element as a child of any type of Layout Region. See Section 21.8, "Layout Regions" on page 21-8.

To create a Custom Button, see Section 31.3.4.18 on page 31-18.

21.9.9 Spacer

Use this element to fine-tune the layout of a page by adding space between other UI elements. For example, create a Spacer element to insert a fixed amount of space within a Layout Region.

When editing a UI, you can:

- Associate a Spacer with any type of Model node
- Create a Spacer as a child of any other UI element.

When editing a UI, you can create a Spacer element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region
- Case Region
- Content Container
- Item Selection Table
- Instance Management Control

To create a Spacer element, see Section 31.3.4.19 on page 31-18.

21.9.10 Separator

This element is useful when you need to visually separate UI content on a Page, such as header text that precedes a group of selection controls. At runtime, a Separator element appears as a thin line below it's parent element. For example, in Figure 21–8 on page 21-15, the lines that appear beneath the text Workstation, CPU, and Monitor is a Separator element.

To create a Separator, see Section 31.3.4.20 on page 31-19.

21.9.11 Check Box

Use this element to display a Model node as a standard HTML check box. At runtime, an end user uses the check box to add an option to or remove it from the configuration (in other words, the element's associated Model node).

You can associate a check box with the following Model structure nodes:

- BOM Model (instantiable or non-instantiable)
- BOM Option Class
- BOM Standard Item
- Option Feature
- Option
- Boolean Feature
- BOM Model Reference

When editing a UI, you can create a Check Box element as a child of the following UI elements:

- Row Layout

- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

To create a Check Box element, see Section 31.3.4.21 on page 31-19.

21.9.12 Enhanced Check Box

This element is a graphical check box that displays the selection state of its associated structure node using the image specified in your UI Master Template. (A node's selection state determines which image appears at runtime.)

For more information, see:

- Section 5.3.1, "Selection State" on page 5-7
- Section 20.2.2.7.1, "Enhanced Check Box and Radio Button Images" on page 20-10

When editing a UI, you can create an Enhanced Check Box element as a child of the same UI elements as those listed in Section 21.9.11, "Check Box" on page 21-22.

To create an Enhanced Check Box element, see Section 31.3.4.22 on page 31-19.

21.9.13 Instantiation Check Box

Use this element to control the instantiation of a Model Reference or a Component that is defined as an Optional Single Instance.

For example, you create this element and, for its Associated Model Node setting, specify a Component whose Instances setting is set to Optional Single Instance. At runtime, an end user selects this check box to create an instance of the Component and add it to the configuration.

This element is part of the predefined Instance Management Control content template. For details about this template, see Section 20.3.3.3, "Instance Management Control Templates" on page 20-17.

When editing a UI, you can create an Instantiation Check Box element as a child of an Instance Management Control or any type of Layout Region. See Section 21.8, "Layout Regions" on page 21-8.

To create an Instantiation Check Box, see Section 31.3.4.23 on page 31-20.

21.9.14 Radio Button

Use this element to display a single Feature Option or BOM item. At runtime, an end user uses the radio button to add an item to, or remove it from, the configuration. (The item referred to here is the element's associated Model node.)

You can associate a Radio Button element with the following Model structure nodes:

- BOM Model
- BOM Option Class
- BOM Standard Item
- Option

When editing a UI, you can create a Radio Button element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container
- Instance Management Control

To create a Radio Button element, see Section 31.3.4.24 on page 31-20.

21.9.15 Enhanced Radio Button

This element is a graphical radio button that displays the selection state of its associated structure node using the image specified in your UI Master Template.

For more information, see:

- Section 5.3.1, "Selection State" on page 5-7
- Section 20.2.2.7.1, "Enhanced Check Box and Radio Button Images" on page 20-10

When editing a UI, you can create an Enhanced Radio Button element as a child of the same UI elements as those listed in Section 21.9.14, "Radio Button" on page 21-24.

To create an Enhanced Radio Button element, see Section 31.3.4.25 on page 31-21.

21.9.16 Drop-down List

Use this element to display a List of Options Feature with a Maximum Selections of 1, or a BOM Option Class containing mutually exclusive Items when displaying the item's quantity is not required.

You can associate a Drop-down List element with the following Model structure nodes:

- BOM Model
- BOM Option Class

- Option Feature

When editing a UI, you can create a Drop-down List element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table
- Instance Management Control
- Instance Management Table
- Item Selection Table
- Connection Targets Table

To create a Drop-down List element, see Section 31.3.4.26 on page 31-21.

21.9.17 Text Input

Use this element to display and set the value of the following Feature types: Text, Integer, Count, Decimal. You can also use it to display the quantity of a BOM item or Counted Option, or the name of a Component or Model instance.

You can associate a Text Input element with the following Model structure nodes:

- BOM Model
- BOM Option Class
- BOM Standard Item
- Text Feature
- Integer Feature
- Decimal Feature
- Instantiable Component

When editing a UI, you can create a Text Input element as a child of the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region

- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

To create a Text Input element, see Section 31.3.4.27 on page 31-22.

21.9.18 Selection Status and Unsatisfied Status Indicators

Use the Status Indicator and Unsatisfied Status Indicator elements to display an image that indicates the status of a specific node at runtime. The settings in the Images section of your UI Master Template control which images appear at runtime.

When you create this element, you specify an associated Model node. At runtime, Oracle Configurator checks the node's selection state and displays the corresponding image specified in your UI Master Template.

For example, you associate the Status Indicator with Feature X and set the Indicator Type to Selection State. When an Oracle Configurator end user selects Feature X, the image that is specified for the User Selected setting in your UI Master Template appears next to Feature X.

For more information, see:

- Section 20.2.2.7.2, "Status Indicator Images" on page 20-10
- Section 5.3.1, "Selection State" on page 5-7

You can associate a Status Indicator element with the following Model structure nodes:

- BOM Model
- BOM Option Class
- BOM Standard Item
- Option Feature
- Option

You can associate an Unsatisfied Indicator element with all of the above except Options. You can also associate this element with Connectors and Text Features. The following nodes can be unsatisfied at runtime: Option Features, BOM Option Classes, Connectors, and Text Features.

When editing a UI, you can create a Status Indicator or Unsatisfied Indicator as a child the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region

- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

To create a Status Indicator or Unsatisfied Indicator element, see Section 31.3.4.28 on page 31-22.

21.10 Other User Interface Elements

The elements described in this section appear in the "Other" category when you are creating a UI element in the User Interface area of the workbench.

For a description of elements that appear in the "Basic" category, see Section 21.9, "Basic User Interface Elements" on page 21-17.

21.10.1 UI Template Reference

For details about this element, see:

- Section 21.16.1, "User Interface Template References" on page 21-48
- Section 21.16, "Generating and Reusing User Interface Content" on page 21-47

21.10.2 Switcher and Case Regions

The Switcher and Case regions are elements you can create in the User Interface area of the Workbench and appear a UI's structure. However, they do not appear in the runtime UI. You use these regions to define a conditional expression that controls which of the Case region's child UI elements appear at runtime.

You specify the condition to evaluate as a runtime property of the Switcher region's associated node (for example, logic state), or the status of the configuration itself (in other words, a configuration session property). You then specify each of the possible results in one or more Case regions. At runtime, Oracle Configurator tests the condition and displays the content defined for the Case region that matches the condition.

For example, a Feature called Custom Car represents a specific kind of automobile, and its three Options represent the available colors. You want to display a different image at runtime based on which Option (color) the end user selects. To do this, you create a Switcher region and select the Custom Car Feature as its associated Model node, and select Selection as the Property. You then create three Case regions as children of the Switcher region (one for each of the Features Options). You then create an Image element as a child of each Case region and specify a different image file for each. Depending on which Option the end user selects, one of the Case conditions evaluates to True and the UI displays corresponding image (that is, the Case's child element).

Whether an item appears may be based on the status of a Model node or the status of the configuration itself. You can associate a Switcher or Case region with any type of Model node.

For example, if the object of the Switcher region (condition) is a structure node, select one of the following System Properties:

- SelectionState
- LogicState
- MinSelected
- MaxSelected
- Satisfied
- SelectableChildren
- Selection
- HasChildren

For more information, see Section 5.3, "System Properties" on page 5-2.

If the object of the Switcher condition is Session Data, select one of the following:

- ModelQuantity
- TotalListPrice
- TotalSellingPrice
- ListPriceEnabled
- SellingPriceEnabled
- PricingEnabled
- ATPEnabled
- PriceAndATPDisabled
- Currency
- Valid
- Unsatisfied
- ConfigHeaderID
- ConfigRevisionNumber
- InNestedTransaction
- IsContainerModel

These properties refer to the status of the configuration itself. For example, if you select Valid as the Session Data property and the configuration is valid, then the contents of the Case region are displayed.

For more information, see Section 5.4, "Configuration Session Properties" on page 5-8.

When editing a UI, you can create a Switcher Region as a child the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout

- Header Region
- Case Region
- Content Container
- Summary Table
- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

You can create a Case Region only under a Switcher Region.

To create a Switcher or Case Region, see Section 31.3.4.30 on page 31-23.

21.10.3 Content Container

A Content Container is a separate region in a UI Page that is offset by a different color. You may want to create this UI element to highlight specific information in a UI page.

For example, you want to display additional information about the item being configured to guide an end user's selections. You create a Content Container containing the word "Tip" as header text and create a Text element to display the text that the end user sees (that is, the container's content).

When editing a UI, you can create a Content Container element as a child the following UI elements:

- Row Layout
- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container
- Instance Management Control

To create a Content Container, see Section 31.3.4.29 on page 31-23.

21.10.4 Summary Table

This UI element displays all orderable options that are selected in a configuration session in a table. You may want to add this element to specific UI pages, for example, so end users can view all selected items without navigating to the Summary page. For more information, see Section 19.5, "The Configuration Summary Page" on page 19-11.

You can create a Summary Table on any UI page, but its associated Model node is always the Model's root node.

When editing a UI, you can create a Summary Table element as a child the following UI elements:

- Row Layout

- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container

See Section 31.3.4.8, "Creating a Summary Table" on page 31-12.

21.10.5 Item Selection Table

An Item Selection Table enables Oracle Configurator end users to select, deselect, and set the quantity of Feature Options or BOM items. Configurator Developer uses these tables when you generate a UI that is based on the Model structure and your UI Master Template uses the Content Templates described in the following sections:

- Section 20.3.3.1, "BOM Item Table Control Templates" on page 20-17
- Section 20.3.3.4, "Counted Option Table Templates" on page 20-18

You can create this type of table manually, or by using one of the templates listed above. Refer to the following sections for more information:

- Section 31.3.4.5, "Creating an Item Selection Table" on page 31-9
- Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7

You can associate an Item Selection Table with the following Model structure nodes:

- BOM Model
- BOM Option Class
- Option Feature

When editing a UI, you can create an Item Selection Table as a child of the following UI elements:

- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container

To create an Item Selection Table, see Section 31.3.4.5 on page 31-9.

21.10.6 Instance Management Table

Create this element to enable an end user to configure and add instances of a Component or Reference to a non-imported Model that can have multiple instances. For background information, see Chapter 7, "Instantiation".

An example of this table is shown in Figure 20–4 on page 20-18.

You can associate an Instance Management Table with the following Model structure nodes:

- Instantiable Component
- Instantiable Model Reference

When editing a UI, you can create an Instance Management Table as a child of the following UI elements:

- Cell Format
- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container

You can create this table manually, or use the predefined Instance Management Table UI Content Template to generate it. For details, see:

- Section 31.3.4.6, "Creating an Instance Management Table" on page 31-10
- Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7

Note: If your UI uses a Dynamic Model Tree navigation style and you specify Sorting settings for an Item Selection Table, the order in which instances appear in the Tree might not reflect the order of instances in the table. The Sorting settings are described in Section 31.3.13, "Sorting Options" on page 31-29.

21.10.7 Connection Navigator Table

Create this element to list all components that are currently connected to the selected instance in a table at runtime. (In other words, all components that are connected to the component the Oracle Configurator end user is viewing.) This element is useful in Models that support connectivity. For details, see Chapter 8, "Connectivity".

When you create this element using the Connection Navigator Template, each component name appears as a link that the end user can use to navigate to that component. When you create a Connection Navigator Table manually, you must create each link separately. For details about the Connection Navigator Template, see Section 20.3.3.13 on page 20-20.

A Connection Navigator Table is associated with the root Model node by default, and you cannot change this association. For details about associated Model nodes, see Section 21.14 on page 21-43.

A Connection Navigator Table does not contain information at runtime if no connections exist in the configuration.

When editing a UI, you can create a Connection Navigator Table as a child of the following UI elements:

- Cell Format

- Flow Layout
- Stack Layout
- Table Layout
- Header Region
- Case Region
- Content Container

For more information, see:

- Section 31.3.4.7, "Creating a Connection Navigator Table" on page 31-11
- Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7

21.10.8 Navigation Bar

Use this element to indicate which Page in a sequence the end user is viewing. For example, when you add this element to each UI Page in a Page Flow containing five Pages, this element displays "Page 1 of 5" on the first Page, "Page 2 of 5" on the second, and so on.

This element is part of the following predefined UI Content Templates:

- Section 20.3.2.4, "Step-by-Step Navigation Bar" on page 20-16
- Section 20.3.2.5, "Two-Page Navigation Bar" on page 20-16

21.10.9 Raw Text

Use this element to add an HTML directive to a UI. This element displays unescaped HTML, which means the runtime Oracle Configurator does not scan the text for "special characters" and substitute them so they show up as entered in an HTML page.

For example, to display the text "Go to our Web site" at runtime as a link to www.ourwebsite.com, enter the following:

```
<A href='www.ourwebsite.com'>Go to our Web site</A>
```

When editing a UI, you can:

- Associate a Raw Text element with any type of Model node
- Create a Raw Text element as a child of any other UI element.

When editing a UI, you can create an Raw Text element as a child of the following UI elements:

- Flow Layout
- Stack Layout
- Bulleted List
- Table Layout
- Header Region
- Case Region
- Content Container
- Summary Table

- Item Selection Table
- Instance Management Table
- Instance Management Control
- Connection Targets Table

To create a Raw Text element, see Section 31.3.4.14 on page 31-16.

21.11 Runtime Conditions and User Interface Elements

You can define one or both of the following types of runtime conditions for any UI element:

- Enabled Condition: When the condition you define is true at runtime, the UI element is disabled.
- Display Condition: When the condition you define is true at runtime, the UI element and its child content (if any) no longer appear.

The condition you define affects both the selected UI element and all of its children. For example, when a condition causes a UI page to be disabled, all of the page's content and controls are visible, but are read-only (this does not include navigation controls such as a Menu or Step-by-Step navigation buttons).

An element may be hidden or disabled at runtime based on a Model node's System or User Property, or a property of the configuration session itself (for example, Valid or Unsatisfied). For example, you may want a button to be disabled when the logic state of a specific node is Logic False, or hide a Configure button when its associated component is not instantiable. For more information about each type of Property, see Chapter 5, "Properties".

When creating or modifying a Drop-down List, Node List Layout Region, or any Table element (Summary Table, Item Selection table, and so on), you can define a row or item display condition to dynamically hide specific rows or items in a list. The display condition can be based on either a Property or the state of the row or item's associated Model node. For example, you may want to hide a row when the option it contains is logically excluded from the configuration.

Example 21–2 shows a display condition defined for a user-created button whose action is Update Prices.

Example 21–2 Display Condition for an Update Prices Button

```
Object: Configuration Session
Property: PricingEnabled
Comparison: Is
Condition: True
```

In this example, the Update Prices button does not appear if pricing is not enabled at runtime.

To define a condition for hiding or disabling a UI element, see Section 31.3.10 on page 31-27.

You can also define display conditions for UI elements by creating Switcher and Case Regions. For more information, see Section 21.10.2 on page 21-27.

Note: Defining a display condition for an element that is a child of a Cell Format or a Row Layout is not recommended, as this can produce unexpected results at runtime. Instead, make the element a child of a Flow Layout, and then define a display condition for it.

21.11.1 Hidden User Interface Pages

At runtime, and end user cannot navigate to a UI Page that is hidden because of a display condition. For example, if the UI provides Previous and Next navigation buttons and some UI Pages in the sequence are hidden, Oracle Configurator automatically skips them and displays the next available page in the sequence.

For more information, see Section 19.3, "Controlling the Content of a User Interface" on page 19-10.

21.11.2 Conditions and Effectivity

A UI element's effectivity at runtime takes precedence over any display conditions that you define. For example, when an element is not effective on the day an end user is configuring the product, the element does not appear in the UI, regardless of any display conditions that may defined for that element.

For more information, see Chapter 6, "Effectivity".

21.11.3 Selecting a Valid Object

By default, a display or enabled condition is based on the UI element's associated Model node. However, you can select a different node (the Object in the expression) when defining a condition. When you do this, Configurator Developer ensures that the node you specify is a valid selection. The node you select must be within the mandatory structure of the UI Page's associated Model node, or the mandatory structure of any of its ancestors. In other words, the node cannot be an optional or instantiable descendant of the associated node of the UI Page in which the element appears.

For more information about optional and instantiable components, see Section 11.6, "Rules that Relate Components and Models" on page 11-11.

21.11.4 Problems When Evaluating Runtime Conditions

At runtime, if certain problems occur in evaluating a display or enabled condition, then Oracle Configurator ignores the condition, displays or enables the affected UI element, and presents a notification to the end user.

Evaluation problems include the following:

- The CIO method or other API required to evaluate the runtime condition is not available.
- The node specified by the runtime condition has been deleted from the configuration.
- The value of the Property specified by the runtime condition could not be determined by Oracle Configurator.
- The User Property specified by the runtime condition does not exist.

21.12 User Interface Element Captions and Details

The settings in a UI element’s details page determine how the element appears and functions at runtime. The settings that are available depend on the element’s type, and may include the following:

- **Text Source**: This setting controls how the runtime Oracle Configurator creates an element’s UI caption at runtime. Depending on the type of element, the UI caption can be one of the following:
  - One of the associated Model node’s System Properties

    For example, if you select the `Description` System Property, the element’s UI caption is the description of its associated Model node. System Properties are described in Section 5.3 on page 5-2.
  - One of the associated Model node’s User Properties

    See Section 5.2, "User Properties" on page 5-1.
  - A configuration session property

    For example, if you select the Unsatisfied configuration session property, the element’s UI caption is False when the configuration is complete and True when it is not.

    See Section 5.4, "Configuration Session Properties" on page 5-8.
  - A Text Expression (that is, any text that you enter)

    For example, "Select this box if you are an experienced user." See Section 21.12.1, "Defining a Text Expression" on page 21-36.

  The default value for an element’s UI caption the associated Model node’s `DisplayName` System Property. The value of this Property is derived from a Model-level setting that you specify in the General area of the Workbench. For details, see Section 28.9, "Runtime Display Names" on page 28-6.

  If you select a User Property as the Text Source for a UI element that displays its associated Model node’s children, the Property you select must be common to all of the child nodes. For example, if the UI element is a Drop-down List and the associated Model node is an Option Feature, then the User Property that you specify must be shared by all of the Feature’s Options; otherwise, Configurator Developer displays an error when you save the UI element. For more information about Properties, see Chapter 5, "Properties".

  > **Tip:** If a UI element does not have the Text Source setting, you can create a UI caption for it by defining a Styled Text or Static Styled Text element. See Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14 or Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14.

- **Image Source**: This setting indicates the location of the image file to display at runtime and is available only for Image and Image Button elements. Select one of the associated node’s Properties that is an image file, or select Text Expression and then enter the location of the image file to use.

  For example, specify a complete path to the image you want to use:

  `http://www.MyWebSite.com/image1.gif`

  For more information, see Section 20.2.2.7, "Images Section" on page 20-9.

- **Caption Style** or **Style**: This setting indicates the style sheet definition you want to use to render the UI caption. For example, the style sheet BLAF.xss determines the font, size, and color of the text. This style sheet is located in the `oa_html/cabo/styles` directory.

  For information about personalizing style sheets, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

- **Indicator Type**: This setting controls when a Status Indicator UI element is displayed at runtime. This setting is available only when creating or modifying a Status Indicator element. Choose **Selection State**, **Logic State**, or **Unsatisfied Item**.

  For details about Status Indicators, see Section 21.9.18 on page 21-26.

- Choose a **Rollover Text Source** to specify the node Property, Configuration session property, or text to use as the element's rollover text at runtime. Rollover text indicates a UI element's purpose or action, and appears when an end user places the cursor over the element at runtime, or navigates to an element by pressing the Tab key. For example, rollover text for a **Save** button might be "Click to Save Your Work". For details about defining a text expression, see Section 21.12.1 on page 21-36.

- **Link Action**: Use this setting to assign a runtime action to the selected UI element. For more information, see Section 21.13, "User Interface Actions" on page 21-38.

21.12.1 Defining a Text Expression

Select **Text Expression** as a UI element's Caption Source if you want to define the caption or rollover text. This setting enables you to directly enter the text that appears at runtime for a UI element.

You can also use any System or User Properties that belong to a UI element's associated node when building an expression. When used in a text expression, all Properties must be preceded with an ampersand (&) so the runtime Oracle Configurator will derive the text from the specified Property, rather than displaying the name of the Property itself.

When using a User Property in a text expression, use the following syntax (not case-sensitive):

&PROPERTY["*UserPropertyName*"]

To use the name of a System Property in a text expression, enter the name of the Property and precede it with an ampersand. For example: `&DISPLAYNAME`, `&DESCRIPTION`, or `&QUANTITY` (also not case-sensitive). For a complete list of the System Properties you can use, see Table 5–1 on page 5-2.

For example, you associate a UI element with Feature A and then enter the following expression:

```
You have selected &QUANTITY of this item. Each item adds
&PROPERTY["Weight"] to the configuration.
```

At runtime, Oracle Configurator checks the current value of the associated node's `QUANTITY` System Property and the User Property called "Weight" and displays the following message with Feature A:

```
You have selected 12 of this item. Each item adds 10 ounces to
the configuration.
```

Note: To define an expression that includes the ampersand character at runtime, precede the ampersand with the backslash character. For example, to display "A & B" at runtime, define the following text expression:

```
A \& B
```

The left and right brackets ([and]) are also special characters. If a User Property name contains one of these characters and you want to use the Property name in a text expression, you must precede the bracket with the backslash character. For example:

```
\[DVD RW Player\]
```

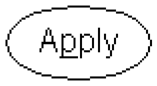
21.12.2 Text Expressions and Keyboard Shortcuts

You may want to define a text expression that includes the keyboard shortcut an end user can to execute an action. For example, you create a Button element and enter the following text expression to serve as its UI caption.

```
A\&pply
```

At runtime, the button appears as shown in Figure 21–9, and the end user can execute the button's action by pressing Ctrl+p.

Figure 21–9 Apply Button with Keyboard Shortcut



To display a text string at runtime that includes an ampersand, you must indicate that you do not want the character to denote a keyboard shortcut. To do this, "escape" the character by preceding it with another ampersand. For example, to display the string "A & B" at runtime, enter the following in the Text Expression field:

```
A \&\& B
```

Table 21–1 lists the common actions defined in Oracle Applications (OA), their corresponding actions in a runtime Oracle Configurator, and the access keys for each.

Table 21–1 Common Runtime Actions and Access Keys

OA Action	Oracle Configurator Action	Access Key (English)
Apply	Apply/Finish	p
Back	Go to Previous Page	k
Cancel	Cancel, Message Decline	l
Continue	Go to Next Page (when Page Flow contains two Pages)	c
Finish	Apply/Finish	i
Next	Go to Next Page (when Page Flow contains more than two Pages)	x
No	Message Decline	n
Yes	Message Confirm	y

Table 21–1 (Cont.) Common Runtime Actions and Access Keys

OA Action	Oracle Configurator Action	Access Key (English)
Save	Save	s

21.13 User Interface Actions

You can assign a runtime action to a Custom Button, Image Button, or a Text Link element. If you want text that has an associated action to be underlined at runtime (so it is apparent that it is a hypertext link), use a Static Styled Text element and enter either OraLink or OraVLinkText for the element's Style setting. For more information about the Style setting, see Section 21.12 on page 21-35.

Assigning an action to a UI element is described in Section 31.3.15 on page 31-30.

Refer to the following sections for details about the available actions:

- Table 21–2, " Navigation Actions" on page 21-38
- Table 21–3, " Model Interaction Actions" on page 21-40
- Table 21–4, " Session Control Actions" on page 21-40
- Table 21–5, " Display Update Actions" on page 21-41
- Section 21.13.1.2, "Raise Command Event Action" on page 21-42

Table 21–2 Navigation Actions

Action	Description
Start Page Flow	Goes to the first UI Page in the Page Flow you specify. For details, see Section 21.6, "Page Flows and Page References" on page 21-7.
Go to Previous Page	Navigates to the previous UI Page in a Page Flow. For more information, see Section 21.15.2, "UI Page Scope and Runtime Navigation" on page 21-45.
Go to Next Page	Navigates to the next UI Page in a Page Flow. The order of UI Pages can change at runtime as components become instantiated or when effectivity and display conditions cause some UI Pages to be hidden. For more information, see Section 21.15.2, "UI Page Scope and Runtime Navigation" on page 21-45.
Go Back	Returns to the page from which the current page was invoked.
Go to Preview Page	Navigates to the Configuration Summary page. For details, see Section 19.5, "The Configuration Summary Page" on page 19-11.
Go to Page	Navigates to the UI Page that you specify. The UI Page you specify as the target of this action must have an associated Model node within the current component's mandatory structure, or the mandatory structure of one of its ancestors. In other words, you cannot select a UI Page whose associated Model node is an optional or instantiable descendant of the current UI Page's associated Model node. For these cases, use the Configure Associated Subcomponent action. For more information, see Section 21.15.3, "UI Page Scope and Instance Navigation" on page 21-45.

Table 21–2 (Cont.) Navigation Actions

Action	Description
Go to Home Page	Navigates to the first page in the UI. The first page depends on the structure of the Model at runtime, which may be affected by effectivity, display conditions, and so on.
Go to Full Summary	Navigates to the Preview Configuration (Summary) page that shows all items in the configuration and displays an icon next to items that have been added, removed, or have changed relative to Oracle Install Base. This action applies only to Models that support updates to installed configurations. For details, see Section 20.3.5.2, "Summary Page Templates" on page 20-22.
Go to Filtered Summary (Configuration Upgrades Only)	Navigates to the Preview Configuration (Summary) page that shows only items (and their parents) that have changed relative to Oracle Install Base. This action applies only to Models that support updates to installed configurations. For details, see Section 20.3.5.2, "Summary Page Templates" on page 20-22.
Go to First Instance	Navigates to the first UI Page for the first instance of the specified Component or Model in the current context. For more information, see Section 21.15.3, "UI Page Scope and Instance Navigation" on page 21-45.
Go to Next Instance	Navigates to the first UI Page for the next instance of the Component or Model that constitutes the current context. For more information, see Section 21.15.3, "UI Page Scope and Instance Navigation" on page 21-45.
Go to Previous Instance	Navigates to the first UI Page for the previous instance of the Component or Model that constitutes the current context. For more information, see Section 21.15.3, "UI Page Scope and Instance Navigation" on page 21-45.
Go to Target Instance	Navigates to the first UI Page for a particular instance of the specified Component or Model.
Configure Associated Subcomponent	Navigates to the UI Page of the element's associated Model node. (See the Note following this table.) If the associated node belongs to a referenced Model and a Page for the node exists in the referenced UI, this action displays that Page. To navigate to a different Page, use the Go To Page action.
Open URL	Navigates to the URL that you specify using the current browser window, or by opening a new window. For details, see Section 21.13.1.1, "Open URL Action" on page 21-41.
Open Connection Chooser	Opens the Connection Chooser page, which displays a list of Model instances to which a component can be connected at runtime. The list appears when an end user clicks a button to create a connection. For more information see Section 20.3.3.11, "Connection Control Template" on page 20-19.

Note: When you assign the Configure Associated Subcomponent action to a UI element, also define the following enabled condition for the element: "Associated Model Node.InstanceCount is not 0". This is required because an error will occur at runtime if an end user executes the action when no instances of the component exist. See Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

Table 21–3 Model Interaction Actions

Action	Description
Select Node	Selects the node that you specify.
Deselect Node	Selects the node that you specify.
Add Instance	Adds an instance of the component that you specify to the configuration.
Add and Go to Instance	Adds an instance of the component that you specify to the configuration, and navigates the end user to the new instance.
Delete Instance	Deletes the selected instance from the configuration. This action derives the instance set from the current context of the page on which it is displayed. For more information about page context and runtime navigation, see Section 21.15 on page 21-44.
Activate Instance	Makes all of the UI controls on the selected, read-only UI page editable. This action applies only to Models that support reconfiguration of installed instances. For more information, refer to the *Oracle Telecommunications Service Ordering Process Guide*.
Set Connection	Creates a connection between the current component and the selected instance (for example, in a Connection Navigator Table).
Clear Connection	Clears the connection between the current component and the selected instance (for example, in a Connection Navigator Table).

Table 21–4 Session Control Actions

Action	Description
Apply/Finish/Confirm	Commits and exits the current transaction. If the end user is in a nested transaction, this action also navigates to the top-level page. Executing this action from a top-level page or flow terminates the configuration session and saves any pending changes.
Cancel	Cancels the current transaction. If the end user is in a nested transaction, this action also rolls back all selections made at or below the current level. Executing this action from a top-level page or flow terminates the configuration session without saving any pending changes.
Message Confirm	Dismisses the message and performs the action(s) described in the message.
Message Decline	Dismisses the message and does not perform the action(s) described in the message.

Table 21–4 (Cont.) Session Control Actions

Action	Description
Revert to Saved	Prompts for confirmation and, if confirmed, clears all assertions made in the current session and navigates to the starting page. Returns to either the initial or restored state of the configuration, depending on what was loaded.
Save	Saves any pending changes. (See Warning below)

WARNING: Do not assign the Save action to a UI control unless you know that the host application supports it. In other words, be sure that the host application can save the configuration without displaying an error. Additionally, creating a control that saves the configuration on a UI Page that exists within a nested transaction can cause an error at runtime. For details about nested transactions, see Section 20.2.2.2.1 on page 20-5.

Table 21–5 Display Update Actions

Action	Description
Update Prices	Update all dynamic pricing information for selected items (for example, item selling price or total price for all items). For information about how a runtime Configurator updates pricing information, see the *Oracle Configurator Implementation Guide*.
Update ATP	Update Available to Promise dates for selected items.
Update Prices and ATP	Update all dynamic pricing information and ATP dates for selected items.

For an important suggestion about the Display Update actions, see Section 20.5 on page 20-23.

21.13.1 Action Parameters

When you select an Action Type of either Open URL or Raise Command Event, the Choose Action page displays additional settings in the Action Parameters section.

21.13.1.1 Open URL Action

Use these settings to specify the destination URL, the target browser window, and the Javascript window options (if creating a new window). If you do not specify window options, Oracle Configurator creates the new window at runtime using the browser's default settings.

You may want to use this option, for example, to enable an end user to view additional information about an option before adding it to the configuration.

- **Target URL Source**: Select **Text Expression** if you want to specify a URL. If you select this option, enter a complete URL. For example:

  ```
  http://www.MyWebPage.com
  ```

  Select **Associated Model Node User Property** if you want to specify a User Property that has a URL as its Property. For more information, see Section 5.2, "User Properties" on page 5-1.

- **Target Window**: Select **Main Window** to navigate to the URL in the current browser window. For an important consideration when using this setting, Section 21.13.1.3, "Forms-Based Host Applications and Target Window" on page 21-43.

  Select **Child Window** to open a separate browser window when the end user invokes the specified action (such as by clicking a button).

  If you select this option:

  – Optionally enter a text string which acts as an internal identifier for the child window. If you leave this field empty, Oracle Configurator creates a string at runtime to identify the window.

    If you enter text, it must not contain any spaces. For example:

    ```
    Window1
    ```

    You may want to specify a Window Name when a UI Page contains more than one control that opens a separate window, and you want the end user to be able to display multiple child windows at the same time.

    For example, Button1 and Button 2 appear in the same UI Page. Each button specifies a different URL, and the Window Name setting is set to Window1 and Window2, respectively. At runtime, when the end user clicks Button1, a separate window appears and displays the specified Web page. If the end user then clicks Button2 (without closing the child window), the specified Web page appears in a second child window.

    If, however, a Window Name for Button2 is not specified, Oracle Configurator replaces the content of the first child window when Button2 is clicked.

  – Specify options for opening the window at runtime in the **Window Options** field. Use valid syntax for the JavaScript method `window.open()`. For example:

    ```
    {toolbar:0,location:0,directories:0,status:0,menubar:0,scr
    ollbars:0,resizable:1,width:200,height:300}
    ```

    Enter a "0" (zero) to disable an option and a "1" to enable it. In the example above, the window does not display a toolbar, but can be resized.

    Not all of the options listed in this example are required, but including them all is recommended. The text you enter must not contain any spaces.

- **Lock Main Window while Displaying Child**: Select this option to prevent the end user from making changes in Oracle Configurator until the child window is closed.

21.13.1.2 Raise Command Event Action

Select the Raise Command Event action to trigger the Configurator Extension(s) bound to the command string that you enter. For example, assign this action if you want to execute a Configurator Extension when an Oracle Configurator end user clicks a button.

For details about the Output Window Name, Output Window Options, and Lock Main Window while Displaying Child settings, see Section 21.13.1.1, "Open URL Action" on page 21-41.

For more information about Configurator Extensions, see Chapter 17.

21.13.1.3 Forms-Based Host Applications and Target Window

If the runtime Oracle Configurator will be launched from a forms-based Oracle Applications product such as Order Management, and you set the Target Window setting to Main Window, you must add the following HTML markup at the bottom of the target page:

```
<IFRAME HEIGHT="0" WIDTH="0" SRC="/OA_HTML/czJradHeartBeat.jsp"
NAME="HeartBeatFrame" Id="HeartBeatFrame" title=""></IFRAME>
```

If this markup does not exist in the target page, the configuration session may end prematurely.

The Target Window setting is described in Section 21.13.1.1, "Open URL Action" on page 21-41.

For more information, see the section on the heartbeat mechanism in the *Oracle Configurator Installation Guide*.

21.14 User Interface Elements and Associated Model Nodes

A UI element is always associated with a Model node. A UI element may be directly associated with a Model node, or it may inherit the node from its parent UI element in the UI structure. This relationship indicates that the UI element represents the structure node in the runtime UI, or is otherwise bound to it in some way. For example, when a UI element is associated with a Model node, the element can use any of the node's System or User Properties as the source of its UI caption. A node's selection state or logic state can also control whether the associated UI element is enabled or visible in the UI. This is described in Section 21.11, "Runtime Conditions and User Interface Elements" on page 21-33.

This relationship can be created automatically, such as when you generate a UI, or manually, when you create a UI element. When you generate a UI using a Master Template, Configurator Developer creates a UI element for each Model node. This creates the default Model node association for each UI element (in other words, the default value of the element's Associated Model Node setting). A UI element that you create is associated with its parent element's node by default. For example, a Layout Region is associated with a Feature node called Primary Color. When you create a Check Box UI element as a child of the Layout Region, the Check Box's Associated Model Node is set to Inherited from Parent by default. In other words, the Check Box is associated with the Primary Color Feature node.

When creating or modifying a UI element, you can change the default Associated Model Node setting and specify a different node. Configurator Developer allows you to create only valid associations between Model structure nodes and UI elements. For example, when modifying an Drop-down List element, you click Choose Node. When you do this, the Model structure appears, but only nodes that can be associated with a Drop-down List can be selected (the Select column is read-only for all invalid nodes). Changing an element's associated Model node is described in Section 31.3.14 on page 31-30.

You can also associate a UI element with a node that belongs to a referenced Model. You may want to do this, for example, to display a node from the referenced Model on one of the Pages in the parent Model.

Because all UI elements appear on a UI Page at runtime, each element's associated node must be within the scope of the Page's associated Model node. For more information, see Section 21.15, "Associated Model Nodes and Page Scope" on page 21-44.

21.15 Associated Model Nodes and Page Scope

When you are creating or editing a UI element, you can refer to a node if it is within the scope of a UI Page; otherwise, Configurator Developer displays an error when you save your changes. The scope of a UI Page refers to the set of all nodes that are reachable within the "current context" when the Page is displayed at runtime. A Page's associated Model node serves as the "page base", and establishes the current context when that Page is displayed at runtime. A node is considered to be "in the scope" of a Page if it is guaranteed to exist and cannot have multiple instances relative to the page base at runtime.

Every node referred to in a Page must be reachable from the page base. This includes the associated Model node for any element on the Page, nodes that participate in a display or enabled condition, or any node that is used as the target of navigation from the UI Page. There are the following exceptions, however:

- A node that is used as the target of the Go to First Instance action may be instantiable relative to the page base.
- A UI element that represents an "instance set" can be bound to an instantiable child of the page base (for example, an Instance Management Table).

In other words, Configurator Developer ensures that a node is reachable by checking the path through the Model structure from the page base to the node referred to on the Page (for example, in a display condition). This path can go up or down through the structure of the Model. However, if the path from the page base to the node in question descends into an optional or instantiable component, then the node is not reachable at runtime, and is therefore out of the Page's scope. In other words, the node might not exist at runtime (because it is optional) or it may be ambiguous because it has multiple instances.

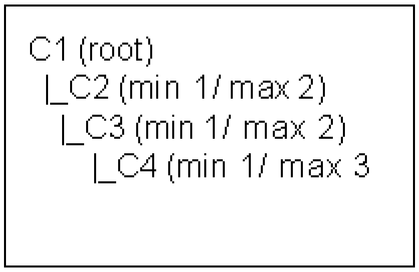
Ascending out of an optional or instantiable component is acceptable, because whenever the descendant structure exists, everything between it and the Model root is guaranteed to exist.

21.15.1 UI Page Scope and Displaying Model Node Content

As explained in the preceding section, all nodes referred to in a UI Page must be within the Page's scope.

Consider the Model structure shown in Figure 21–10.

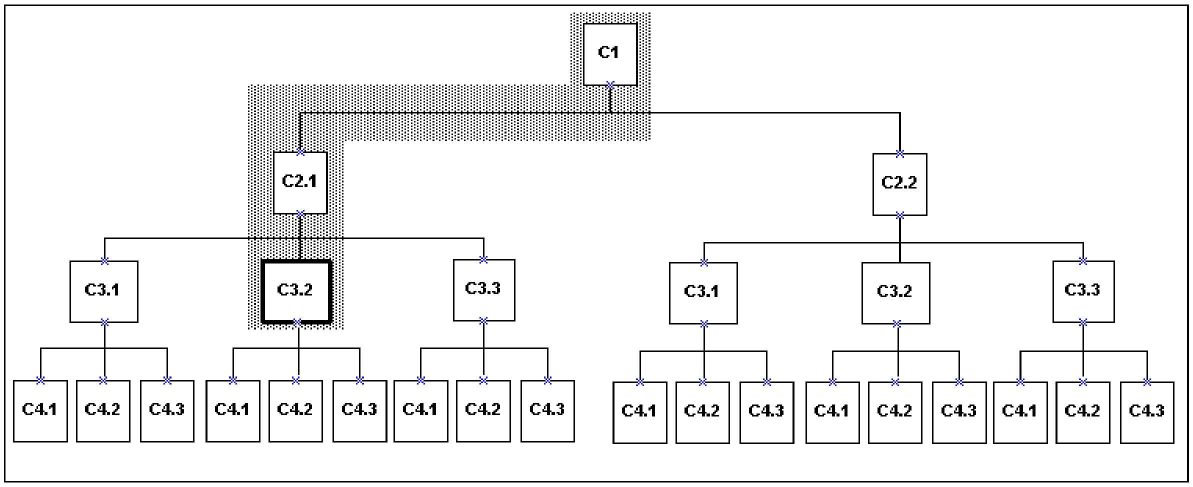
Figure 21–10 Model Structure



```
C1 (root)
 |_C2 (min 1/ max 2)
   |_C3 (min 1/ max 2)
      |_C4 (min 1/ max 3
```

Figure 21–11 shows this Model in its fully-instantiated state at runtime.

Figure 21–11 Current Context and Displaying Model Node Content



In this example, a single UI Page has been generated for each Component. The Page the end user is viewing (C3.2, in bold) is displaying content from the second instance of C3 under the first instance of C2. The Associated Model Node for the current Page is C3, and the shaded area is the scope of the Page. In this context, C3.2 can display content from any of the Component instances within the shaded area, including the Model root.

21.15.2 UI Page Scope and Runtime Navigation

As explained in section Section 21.15 on page 21-44, every node referred to in a UI Page must be reachable from the page base. This includes any node that is used as the target of navigation from the UI Page.

For example, a UI Page must be reachable to be a valid target when you create a button and an assign it to the Go to Page action. That is, the page base of the target Page must be reachable by an unambiguous relative path from the page base of the current page.

In the illustration shown in Figure 21–11 on page 21-45, the component instances that an end user can explicitly navigate to (such as by clicking a button assigned to the Go to Next Page action) include C1 (the root) and C2.1. These nodes both appear in the shaded area, which represents the path mentioned in the preceding paragraphs and in Section 21.15 on page 21-44.

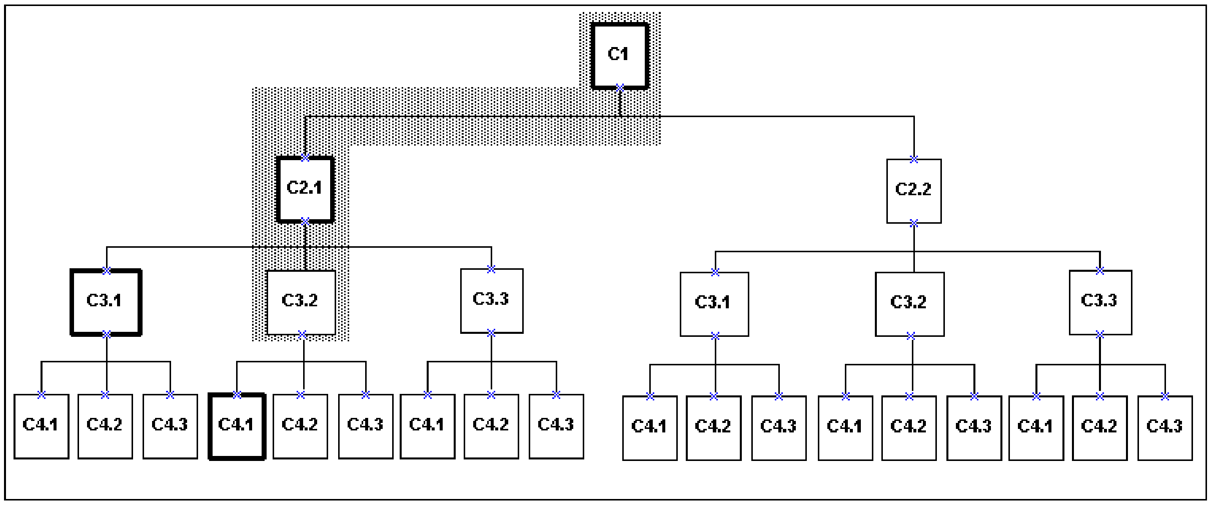
21.15.3 UI Page Scope and Instance Navigation

You can refer to an instantiable Model node as a navigation target using the Go to First Instance action. The Go to First Instance action disambiguates the reference by explicitly referring to the first instance of the specified node. (This means the first instance within a particular component set, not the first instance across multiple sets of the same component.) For example, in Figure 21–11 on page 21-45, an end user can use a UI control assigned to this action to navigate to navigate to C4.1. Therefore, the targeted Model node may be the base component itself (if it is instantiable), the *closest* instantiable descendant of the base component (C4 in Figure 21–11), or any of the base component’s instantiable ancestors (C2 in Figure 21–11).

Navigation by instance always goes to the first Page that is associated with the instantiable component. This is true even if the navigation is invoked from a Page other than the first in another instance.

Figure 21–12 below assumes the same current context as Figure 21–11 on page 21-45, but the base component instances that can be reached by the Go to First Instance action are in bold. In this example these components include the root node, C1, C2.1, C3.1, and C4.1.

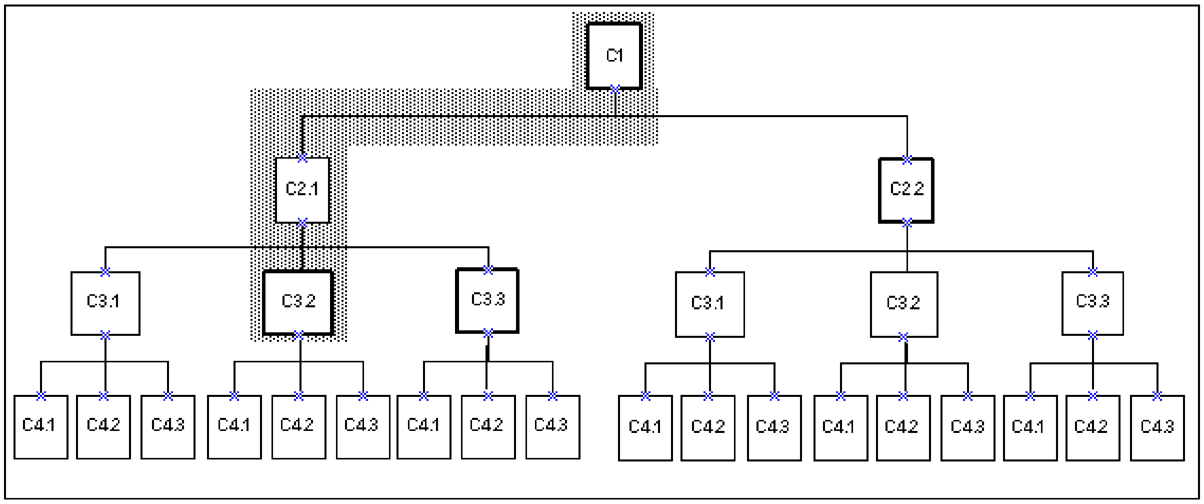
Figure 21–12 Go to First Instance



The target of the Go to Next Instance and Go to Previous Instance actions must also be an instantiable Model node, but unlike the Go to First Instance Action, it must be a Model node that is in the path that constitutes the current context. That is, it must be either the current base component or an instantiable ancestor of the base component. This is because the notion of "next" and "previous" instances implies that the current context defines the "current" instance.

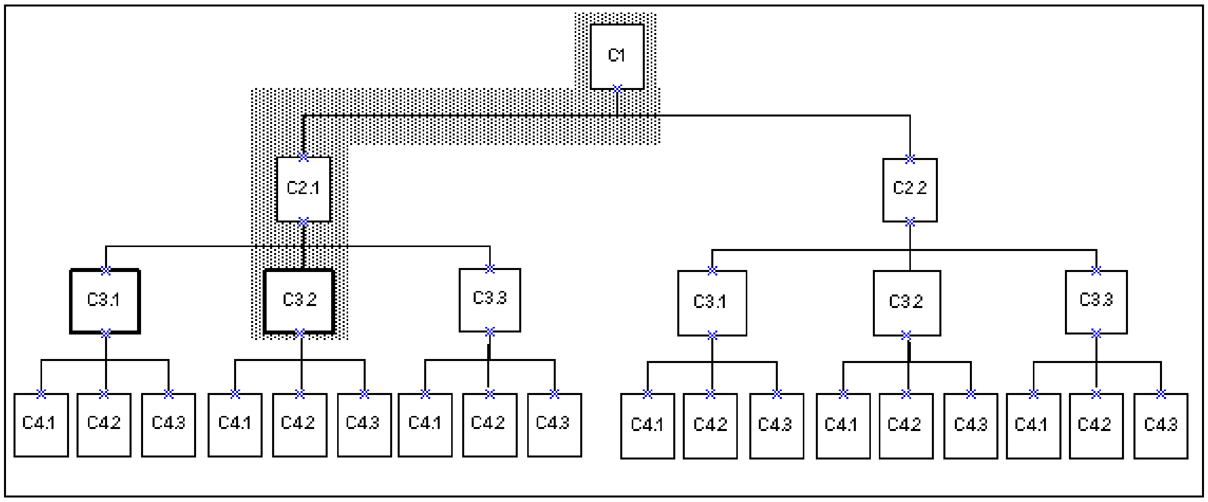
The following illustrations assume the same current context as the previous examples. In Figure 21–13 on page 21-47, the components in bold are the base component instances that can be reached by the Go to Next Instance action from the current UI Page (C3.2). In other words, the root node (C1), C2.2, and C3.3.

Figure 21–13 Go to Next Instance Action



In the Figure 21–14 on page 21-47, the component in bold is the base component instance that can be reached from the current UI Page (C3.2) by the Go to Previous Instance action.

Figure 21–14 Go to Previous Instance Action



For more information, see:

- Section 21.13, "User Interface Actions" on page 21-38
- Section 21.14, "User Interface Elements and Associated Model Nodes" on page 21-43

21.16 Generating and Reusing User Interface Content

When creating or modifying a UI, you can perform the following actions:

- Create a Template Reference: See Section 21.16.1, "User Interface Template References" on page 21-48.
- Create a Region from a Template: See Section 21.16.2, "Creating UI Content from a User Interface Content Template" on page 21-50.
- Convert a Template Reference into UI content: See Section 31.3.2.1, "Converting a UI Template Reference" on page 31-6.

21.16.1 User Interface Template References

If you are unfamiliar with UI templates, review Chapter 20, "User Interface Templates" before reading this section.

A User Interface Template Reference is a UI element that enables a UI to refer to a UI Content Template and display its content dynamically at runtime. In other words, any changes to a UI Content Template appear automatically the next time you unit test the UI.

User Interface Template References can be created manually or automatically. You create a UI Template Reference manually when editing a UI. You may want to do this to associate a Model node with a specific UI Content Template. For example, you want all Option Features in your Model to appear the same way at runtime, so you create an Option Feature UI Content Template. When editing the UI, you create a UI Template Reference for each Option Feature in your Model, and specify your Option Feature template. Since the Option Features reference your template, Oracle Configurator displays the Option Features correctly, even if the template has recently changed.

By default, Template References that you create appear in the UI structure as "UI Template Reference - *Number*." For example: UI Template Reference - 3. See Figure 21–15 on page 21-49.

Creating a Template Reference is described in Section 31.3.4.2, "Creating a UI Template Reference" on page 31-7.

When you create a UI that is based on the Model structure, Configurator Developer automatically creates Template References wherever the Template Usage setting in the UI Master Template is set to Incorporate by Reference. This setting is described in Section 20.3.1 on page 20-14.

For example, in the BOM Content Custom Settings page, the Multi-Instance setting for Instantiable BOM Model References is set to BOM Instance Management Table, and the Template Usage is set to Incorporate by Reference. After you generate the UI, the UI structure contains a Template Reference for each BOM Model Reference that can have multiple instances at runtime. For an example of manually creating and using Template References, see Section 21.17, "Designing and Creating a User Interface Page" on page 21-50.

If necessary, you can convert a Template Reference into UI content. This is described in Section 31.3.2.1, "Converting a UI Template Reference" on page 31-6.

When editing a UI, you can create a Template Reference as a child the following UI elements:

- Row Layout
- Flow Layout
- Stack Layout
- Table Layout
- Header Region

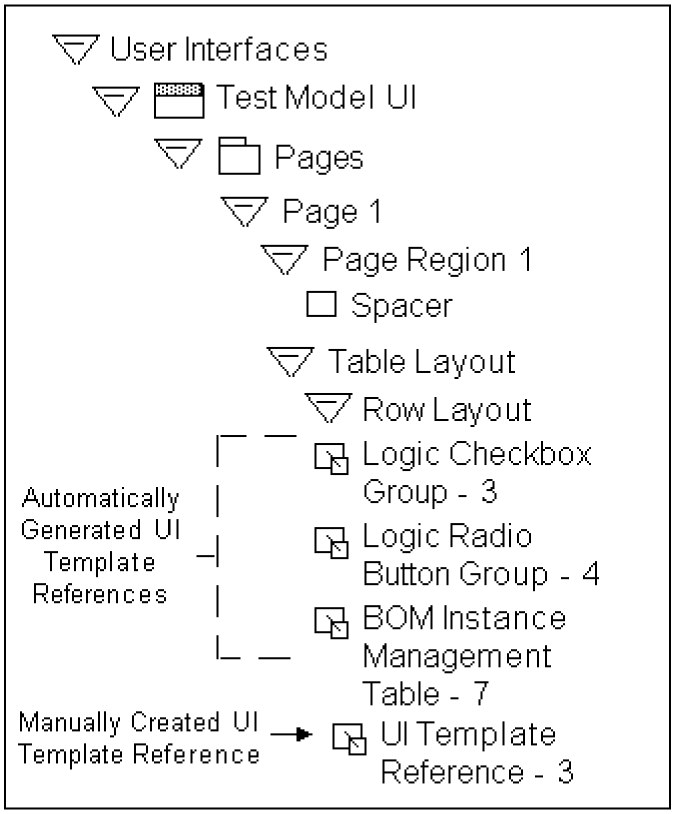
- Case Region
- Content Container
- Instance Management Control

21.16.1.1 Viewing UI Template References

A row in the UI structure that is a reference to a UI Content Template does not have any child elements and contains the name of the template, followed by a sequence number. For example: "BOM Instance Management Table - 7". When viewing the reference's details page, you can see which UI Content Template to which it refers and if necessary, specify a different template. See Section 21.16.1.2, "Modifying a UI Template Reference" on page 21-49.

Figure 21–15 shows UI structure containing several UI Template References.

Figure 21–15 UI Template References


User Interfaces
Test Model UI
Pages
Page 1
Page Region 1
Spacer
Table Layout
Row Layout
Logic Checkbox Group - 3
Logic Radio Button Group - 4
BOM Instance Management Table - 7
UI Template Reference - 3
Automatically Generated UI Template References
Manually Created UI Template Reference

21.16.1.2 Modifying a UI Template Reference

You may want to modify an existing Template reference so it points to another template. When you do this, you may need to make some changes to the enclosing region if the Layout Styles of the old and new templates are different. This is because some UI elements may not appear as expected at runtime when using a different template.

For details, see Section 31.3.4.3, "Modifying a UI Template Reference" on page 31-8.

For example, you may need to move the template out of its old region, or create a new Layout Region to wrap it. You can view a template's Layout Style by editing the template's Definition page. For details, see Section 31.4.3, "User Interface Content Template Settings" on page 31-32.

21.16.2 Creating UI Content from a User Interface Content Template

When creating a UI from scratch or editing a generated UI, you can generate UI elements using a UI Content Template. Creating UI content using a template is an efficient way of creating frequently used or more complex UI elements, such as tables.

When you create an element using a template, Configurator Developer copies the content of the selected template into the UI at the location you specify. You can then modify the UI element's default settings, such as its UI caption, runtime conditions, and so on.

You manually create UI content from a UI Content Template by selecting "Create Region from Template" when creating UI Page content.For details, see Section 31.3.4.1, "Creating a Region from a User Interface Content Template" on page 31-7.

Configurator Developer automatically creates UI content from a UI Content Template when the Template Usage setting in the UI Master Template is set to Copy to UI as Page Content. For details, see Section 20.3.1, "Specifying How a User Interface Uses Content Templates" on page 20-14.

21.17 Designing and Creating a User Interface Page

This section provides an introduction to UI editing and explains how you can use the UI elements that Configurator Developer provides to edit a UI and create a UI Page from scratch.

Before reading this section, you should be familiar with the UI elements described in this chapter and understand how of User Interface templates are used when creating, editing, and refreshing a UI. UI templates are described in Chapter 20.

Creating a custom UI Page typically consists of the following tasks:

- Step 1: Create an Initial Design of the New UI Page
- Step 2: Plan for Using Layout Regions to Arrange the Page's Content
- Step 3: Plan for Using UI Elements and Custom UI Content Templates
- Step 4: Create the UI Page and Page Content
- Step 5: Review and Modify the Page

21.17.1 Introduction to the Example Model

The example Model is called the Windows Model and it was created by a windows manufacturer to enable end users (its customers) to configure and order windows on an external Web site. The Windows Model is based on an imported BOM Model, and the available types of windows are Composite, Vented, Fixed, and CircleHead.

Following are the basic requirements for the UI:

- End users (customers) must be able to specify requirements for each window.

  In the Windows Model, customer requirements include the project type, window material, interior and exterior colors, a glazing type, and a unit of measure.

- End users must be able to create and configure multiple instances of each type of window, specify a quantity for each window, and delete instances.

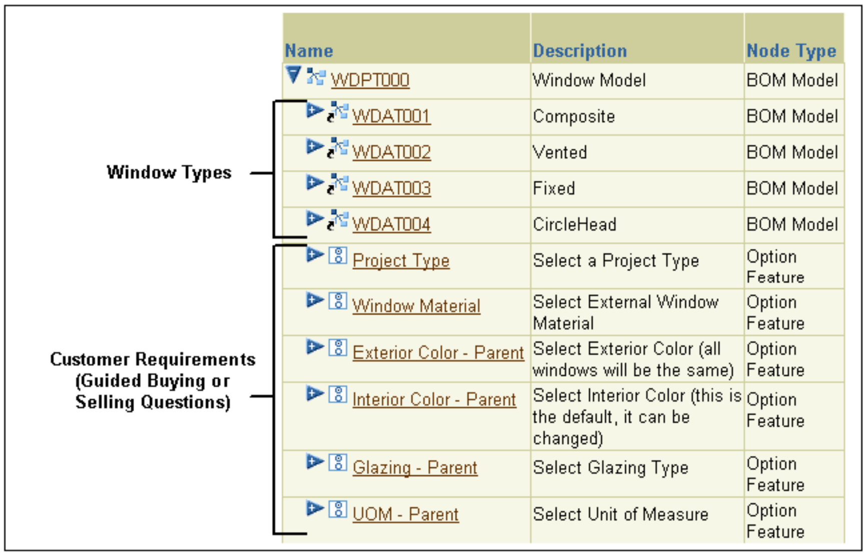
- End users must be able to create instances of a window by clicking on a picture of the window.
- The page must provide a link to additional details about each type of window.
- UI elements should be referenced by the UI wherever possible.

  This is required because the windows manufacturer wants to display text and selection controls in a specific way and reuse these UI elements within other Models and UIs.

In the Windows Model, customer requirements questions are defined using Option Features, while each type of window is a referenced BOM Model that can have multiple instances at runtime.

Figure 21–16 shows the structure of the Windows Model as it appears in Configurator Developer.

Figure 21–16 Windows Model Structure



Name	Description	Node Type
WDPT000	Window Model	BOM Model
WDAT001	Composite	BOM Model
WDAT002	Vented	BOM Model
WDAT003	Fixed	BOM Model
WDAT004	CircleHead	BOM Model
Project Type	Select a Project Type	Option Feature
Window Material	Select External Window Material	Option Feature
Exterior Color - Parent	Select Exterior Color (all windows will be the same)	Option Feature
Interior Color - Parent	Select Interior Color (this is the default, it can be changed)	Option Feature
Glazing - Parent	Select Glazing Type	Option Feature
UOM - Parent	Select Unit of Measure	Option Feature

21.17.2 Step 1: Create an Initial Design of the New UI Page

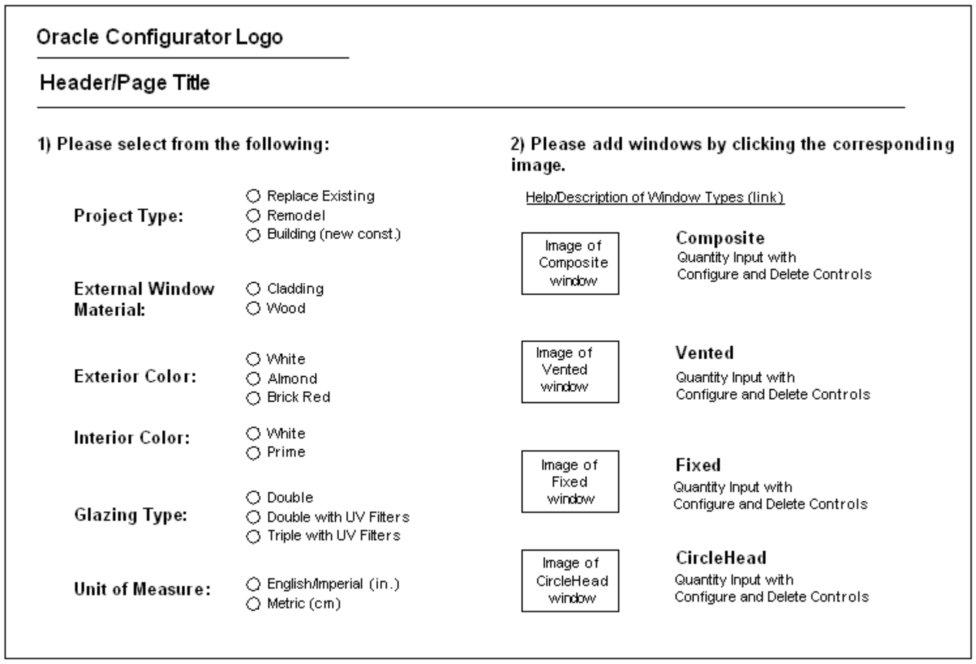
The first step when creating a custom UI Page or editing a UI is to determine what you want to display and how you want the content to be arranged on the Page. In other words, decide how you want the completed UI Page to appear to your end users at runtime.

In this step, carefully consider the features and options of your Model and determine whether you want any guided buying or selling questions (to capture the customer's requirements). Also, identify which product images you want to display. Finally, decide upon the most effective way to present required information to your end users in a format that is easy to understand and use.

Creating a rough sketch of the Page and its content is recommended during this step. For example, you may want to create initial designs using a whiteboard and then refine the layout and content using graphics software.

Figure 21–17 on page 21-52 shows the initial design of the new UI Page for the Windows Model. It includes the page header, the text of each customer requirements question, and the controls required to respond to these questions. It also shows where the window images and the controls for configuring each window instance appear on the Page.

Figure 21–17 UI Page Design - Initial Design


Oracle Configurator Logo

Header/Page Title

1) Please select from the following:

Project Type:
- Replace Existing
- Remodel
- Building (new const.)

External Window Material:
- Cladding
- Wood

Exterior Color:
- White
- Almond
- Brick Red

Interior Color:
- White
- Prime

Glazing Type:
- Double
- Double with UV Filters
- Triple with UV Filters

Unit of Measure:
- English/Imperial (in.)
- Metric (cm)

2) Please add windows by clicking the corresponding image.

Help/Description of Window Types (link)

Image of Composite window — **Composite** Quantity Input with Configure and Delete Controls

Image of Vented window — **Vented** Quantity Input with Configure and Delete Controls

Image of Fixed window — **Fixed** Quantity Input with Configure and Delete Controls

Image of CircleHead window — **CircleHead** Quantity Input with Configure and Delete Controls

After completing the initial design of the UI Page, the next step is to decide which UI elements will be required to arrange the Page’s content.

21.17.3 Step 2: Plan for Using Layout Regions to Arrange the Page’s Content

The initial design of the UI Page, shown in Figure 21–17 on page 21-52, has two main regions. The region on the left consists of several customer requirements questions and responses for each. The region on the right consists of:

- Text that prompts the end user to add window instances
- An image and text link to more information about window types
- Pictures of each window type
- The controls required to configure, delete, and enter a quantity for each window instance

Identifying the Page’s main regions is useful when determining which UI elements will be required to arrange the text, selection controls, and images on the Page. UI elements that arrange other elements on a Page are called Layout Regions. Layout Regions are described in Section 21.8 on page 21-8.

In this step, you may want to create another drawing to visualize the Page’s main areas and the Layout Regions required to create them. Figure 21–18 on page 21-53 shows the Layout Regions and other UI elements that will be required to create the new UI Page based on the initial design.

Figure 21–18 UI Page Design - Required Layout Regions and Other UI Elements

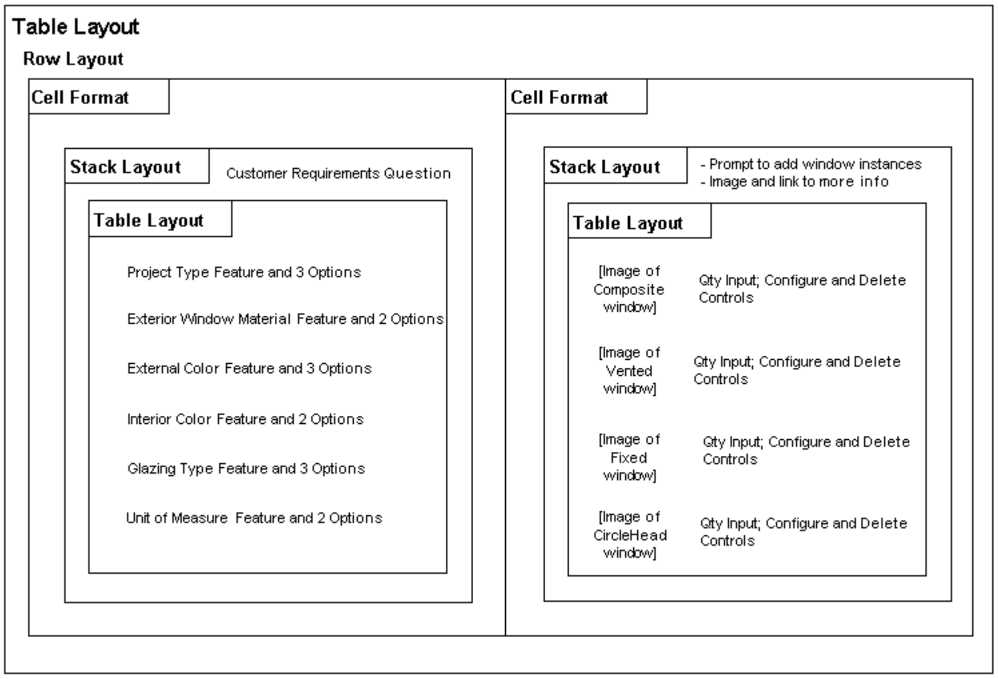


Table Layout
Row Layout
Cell Format
Stack Layout
Customer Requirements Question
Table Layout
Project Type Feature and 3 Options
Exterior Window Material Feature and 2 Options
External Color Feature and 3 Options
Interior Color Feature and 2 Options
Glazing Type Feature and 3 Options
Unit of Measure Feature and 2 Options
Cell Format
Stack Layout
- Prompt to add window instances
- Image and link to more info
Table Layout
[Image of Composite window]
Qty Input; Configure and Delete Controls
[Image of Vented window]
Qty Input; Configure and Delete Controls
[Image of Fixed window]
Qty Input; Configure and Delete Controls
[Image of CircleHead window]
Qty Input; Configure and Delete Controls

Figure 21–18 shows the following:

- The two main regions of the Page are represented by two Cell Formats. Since they must appear side-by-side at runtime, they must be created under a Row Layout in the UI structure.
- The customer requirements questions, the images of each window, and the option selection controls must be arranged vertically within the two main regions. Therefore, these UI elements must be created beneath a Stack Layout.
- The two Table Layouts within the Stack Layouts provide additional control over formatting and ensure that the text, images, and selection controls are properly aligned within the two main regions.

21.17.4 Step 3: Plan for Using UI Elements and Custom UI Content Templates

In this step, decide how you want to display Model nodes at runtime and define any custom UI Content Templates that the UI requires to do this. Also, consider what other kinds of UI elements will be required to create the UI Page according to the initial design.

The UI requirements listed in Section 21.17.1, "Introduction to the Example Model" on page 21-50 state that UI elements must be reusable. This is because the windows manufacturer wants to use the UI controls that provide customer requirements questions and the controls provided to manage instances of each window in multiple Models and UIs. This can be accomplished by defining custom UI Content Templates.

21.17.4.1 Custom UI Content Templates and Template References

The windows manufacturer created two custom UI Content Templates. The ADS - Radio Button Group template displays the customer requirements controls, and the ADS - BOM Instance Management Table template displays controls for configuring each window instance. These templates were created by copying and then modifying the predefined Enhanced Radio Button Group and BOM Instance Management Table templates.

The ADS - Radio Button Group template creates a UI caption using its associated Model node’s description and displays the node’s selectable child options using radio buttons. In the Page for the Windows Model, this template displays the following Features and their Options: Project Type, Window Material, Exterior Color, Interior Color, Glazing, and UOM. References to this template appear in the UI structure shown in Figure 21–19 on page 21-56.

The ADS - BOM Instance Management Table template also creates UI captions (the table headers) using its associated Model node’s description, and it enables end users to enter a quantity, configure, and optionally delete instances of each window. References to this template appear in the UI structure shown in Figure 21–20 on page 21-56.

The Template References that refer to the custom UI Content Templates are shown in Figure 21–20 on page 21-56.

Note: When editing a UI, you reference a UI Content Template by creating a Template Reference and then specifying the node you want the template to represent in the UI as its Associated Model Node (for example, the Project Type Option Feature). For more information, see Section 21.16.1, "User Interface Template References" on page 21-48.

21.17.4.2 Required UI Elements

The requirements for the UI Page also state that end users must be able to create instances by clicking on a picture of each window. The Image Button UI element provides this functionality. This element is described in Section 21.9.6 on page 21-20.

In the Page for the Windows Model, each Image Button:

- Displays a picture of a different window

  The UI element’s Image Source setting identifies the image to display.

- Is associated with a different BOM Model Reference

  These nodes represent the available window types (see Figure 21–16 on page 21-51).

- Has an action of Add Instance

  This enables the end user to create an instance of each window by clicking on the corresponding image.

The UI Page also requires an Image Button to display the image of a question mark that the end user can click on for more information about the available window types. This image will appear next to, and have the same action as, the "details about window types" text link.

21.17.5 Step 4: Create the UI Page and Page Content

In this step you create all of the UI elements that the UI Page requires in the User Interface area of the Workbench. This section assumes that you have already generated a UI that is based on the Model structure, or have created an empty UI. These tasks are described in Chapter 31, "User Interface Area of the Workbench".

Refer to the following sections for details about UI elements that are commonly used when editing a UI or building a custom UI Page:

- Section 21.8, "Layout Regions" on page 21-8
- Section 21.9, "Basic User Interface Elements" on page 21-17
- Section 21.10, "Other User Interface Elements" on page 21-27
- Section 31.3.4.2, "Creating a UI Template Reference" on page 31-7

21.17.5.1 Creating the UI Page and Required Elements

This section explains how to create the left hand region of the UI Page in the User Interface area of the Workbench. The completed structure that represents this part of the UI is shown in Figure 21–19 on page 21-56.

Following are the basic steps to create this part of the UI:

1. In the Pages Folder of the Windows Model UI, create a UI Page. See Section 31.3.3, "Creating a User Interface Page" on page 31-6 for details.

   All of the elements in the following steps will be created under this Page, so they will appear on this Page at runtime.

2. To create a region that includes two side-by-side regions in the UI Page, create a Table Layout element. This new element is named Table Layout 1 in Figure 21–19 on page 21-56 because it is the first Table Layout element in the UI Page.

   See Section 31.3.4.4, "Creating a Layout Region" on page 31-8 for details.

3. Create a Row Layout under the Table Layout. This region will arrange the two main regions of the Page side-by-side at runtime.

   See Section 31.3.4.4, "Creating a Layout Region" on page 31-8 for details.

4. Under Row Layout 1, create a Cell Format. This is the left region of the Page which will contain the customer requirements selection controls.

5. Under the Cell Format you just created, create a Stack Layout element.

   This element will display the customer requirement selection controls vertically at runtime.

6. Create the following elements under the Stack Layout:

   a. A Formatted Text element, which is the text of the prompt for responding to the customer requirements question. The element’s Text Source is set to Text Expression and it contains the following text: "Please select from the following:"

   b. A Spacer element, which adds space between the customer requirements question "Please select from the following:" and the Radio Button groups that appear below it on the Page (see Figure 21–17 on page 21-52). In this example, the value of the Height setting is 25 (pixels).

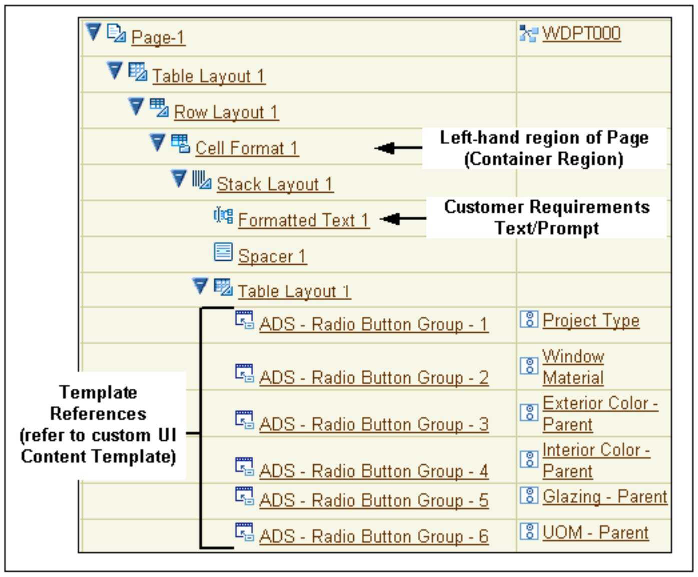
   c. A Table Layout element, which arranges the customer requirements selection controls (Radio Buttons) that you create in the next step.

7. Under the Table Layout, create six Template References.

   All of these Template References refer to the ADS - Radio Button Group template, but each has a different associated Model node. For example, the first is associated with the Project Type Feature, the next is associated with the Window Material Feature, and so on.

8. Following the same procedure, create another Cell Format (Cell Format 2) under Row Layout 1, and then create the elements shown in Figure 21–20 on page 21-56.

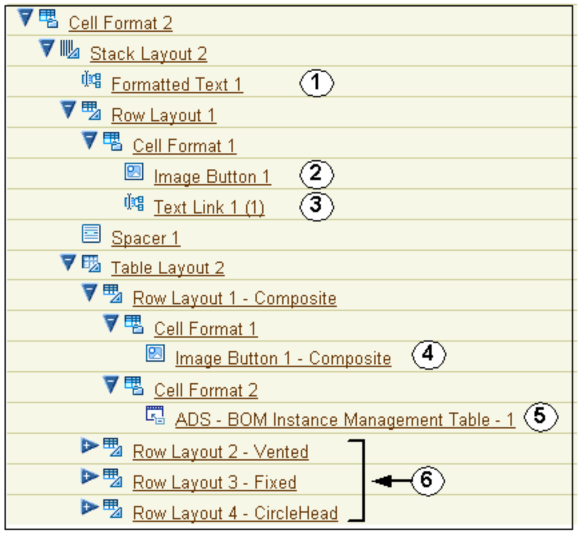
Figure 21–19 Partial UI Structure - Left Hand Region of UI Page



```
Page-1                                          WDPT000
  Table Layout 1
    Row Layout 1
      Cell Format 1
        Stack Layout 1
          Formatted Text 1
          Spacer 1
          Table Layout 1
            ADS - Radio Button Group - 1        Project Type
            ADS - Radio Button Group - 2        Window Material
            ADS - Radio Button Group - 3        Exterior Color - Parent
            ADS - Radio Button Group - 4        Interior Color - Parent
            ADS - Radio Button Group - 5        Glazing - Parent
            ADS - Radio Button Group - 6        UOM - Parent
```

Left-hand region of Page (Container Region)
Customer Requirements Text/Prompt
Template References (refer to custom UI Content Template)

Figure 21–20 Partial UI Structure - Right Hand Region of UI Page



```
Cell Format 2
  Stack Layout 2
    Formatted Text 1                              1
    Row Layout 1
      Cell Format 1
        Image Button 1                            2
        Text Link 1 (1)                           3
    Spacer 1
    Table Layout 2
      Row Layout 1 - Composite
        Cell Format 1
          Image Button 1 - Composite              4
        Cell Format 2
          ADS - BOM Instance Management Table - 1 5
      Row Layout 2 - Vented
      Row Layout 3 - Fixed                        6
      Row Layout 4 - CircleHead
```

Following is a description of the numbered UI elements shown in Figure 21–20.

- Element 1, Formatted Text with the text "Please add windows by clicking on the corresponding image."
- Element 2, Image Button that appears as a question mark at runtime. (See Figure 21–21 on page 21-58).
- Element 3, Text Link that displays the following text at runtime: "More information about window types."
- Element 4, Image Button that enables end user to add instances of the Composite window.

- Element 5, Template Reference that displays a UI control for each instance of the Composite window.

  This Template Reference refers to the ADS - BOM instance Management Table template, and is associated with the Composite BOM Model Reference node.

- Elements 6, Row Layouts 2, 3, and 4 with the same structure as Row Layout 1. The only difference is that the Template References are associated with the Vented, Fixed, and CircleHead BOM nodes (that is, the other window types), and the Image Buttons display images of the Vented, Fixed, and CircleHead windows.

To see how the completed UI Page appears at runtime, see Figure 21–21 on page 21-58.

Note: Row Layouts 1-4 could also be defined as UI Content Templates and then incorporated into the UI as Template References.

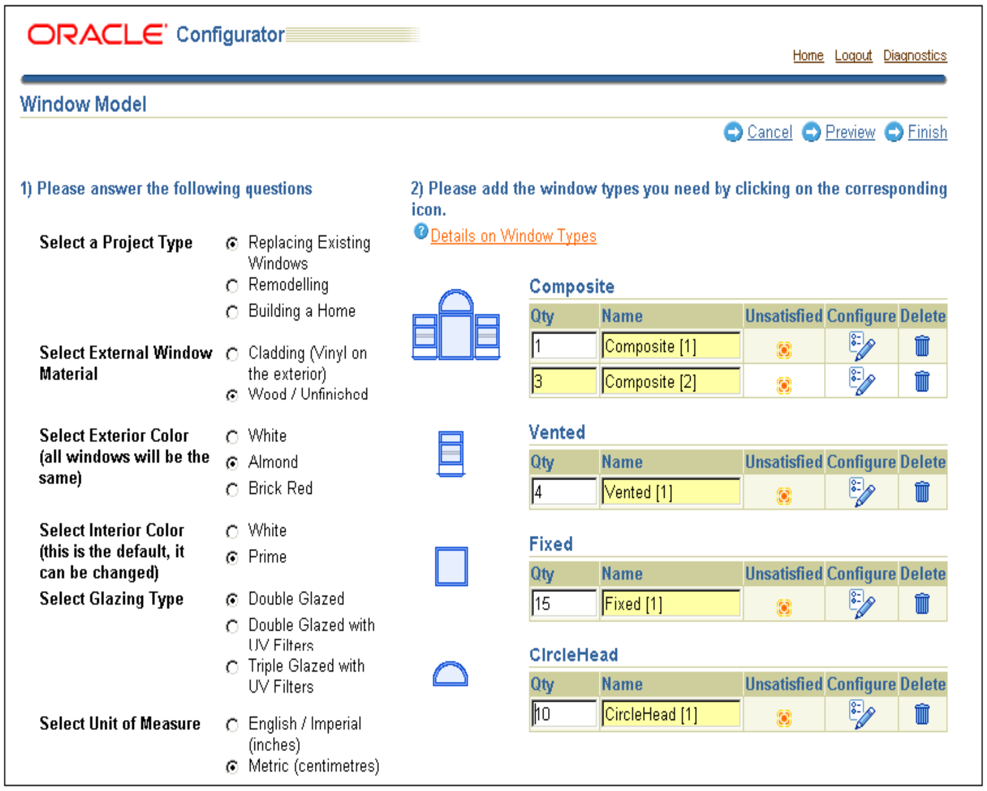
21.17.6 Step 5: Review and Modify the Page

When creating a UI Page or simply editing a UI to make minor changes, it is a good idea to periodically review its appearance at runtime to see the effect of any recent changes. Then, return to Configurator Developer and modify the UI structure as required. For details, see Section 22.3, "Unit Testing a Generated User Interface" on page 22-3.

For example, you may want to change the order in which selection controls appear on the Page, modify existing Layout Regions to enhance alignment of specific elements, or modify a Template Reference to point to a different UI Content Template.

Figure 21–21 shows the completed UI Page for the Windows Model.

Figure 21–21 Completed UI Page at Runtime



ORACLE Configurator

Home Logout Diagnostics

Window Model

Cancel Preview Finish

1) Please answer the following questions

Select a Project Type	[✓ Replacing Existing Windows] Remodelling Building a Home
Select External Window Material	Cladding (Vinyl on the exterior) [✓ Wood / Unfinished]
Select Exterior Color (all windows will be the same)	White [✓ Almond] Brick Red
Select Interior Color (this is the default, it can be changed)	White [✓ Prime]
Select Glazing Type	[✓ Double Glazed] Double Glazed with UV Filters Triple Glazed with UV Filters
Select Unit of Measure	English / Imperial (inches) [✓ Metric (centimetres)]

2) Please add the window types you need by clicking on the corresponding icon.

Details on Window Types

Composite

Qty	Name	Unsatisfied	Configure	Delete
1	Composite [1]	●	✎	🗑
3	Composite [2]	●	✎	🗑

Vented

Qty	Name	Unsatisfied	Configure	Delete
4	Vented [1]	●	✎	🗑

Fixed

Qty	Name	Unsatisfied	Configure	Delete
15	Fixed [1]	●	✎	🗑

CircleHead

Qty	Name	Unsatisfied	Configure	Delete
10	CircleHead [1]	●	✎	🗑

In Figure 21–21 on page 21-58, the end user has responded to the customer requirements questions that appear in the left-hand region of the Page, and an instance of each type of window has been created. The icon in the Unsatisfied column indicates that all of the window instances contain at least one required selection.

Part V

Testing and Publishing

Part V presents information about unit testing and publishing configuration models.

Part V contains the following chapters:

22 Testing and Debugging

This chapter describes how to unit test a configuration model.

This chapter includes the following sections:

22.1 Unit Testing

Unit testing enables you to review and make iterative changes while a configuration model is still in development. Perform unit testing to:

- Review a configuration model and rules using the Model Debugger
- View the Model structure in a UI before creating any configuration rules
- Perform a final check before releasing the Model for testing by external users (for details, see Chapter 23, "Publishing")

Test configuration rules in either the runtime Oracle Configurator or the Model Debugger to verify that they function as intended. It is good practice to test configuration rules incrementally. For example, if you receive an error in the runtime Oracle Configurator, you can temporarily disable one or more rules and then retest to determine which rule is causing the error. Then modify the rule in Configurator Developer to resolve the problem. Disabling rules is described in Section 11.2.4 on page 11-2.

Before unit testing, be sure to generate logic. For details, see Section 28.7 on page 28-5. Additionally, if you plan to test using a generated User Interface, you may need to refresh the UI. For details, see Section 28.8 on page 28-6.

22.1.1 Unit Test Session Environments

You unit test a configuration model from Configurator Developer by launching the Model Debugger, or a User Interface that you generated in Configurator Developer. You can access both unit testing environments from either Configurator Developer or the Oracle E-Business Suite Home page.

In Configurator Developer, you can unit test the Model that is open for editing by clicking the Test Model button in the Structure, Rules, or User Interface area of the Workbench. From the E-Business Suite Home page, launch a runtime UI or the Model

Debugger by selecting Test Configuration from the main menu. In either case, you can enter session parameters before testing.

22.1.1.1 Session Parameters

Session Parameters enable you to apply effectivity and specify a Model quantity when unit testing a configuration model in the Model Debugger or a runtime UI. When you enter an effective date, any Model structure that is not effective for the date or Usage that you specify does not appear, and any ineffective rules are ignored. If you do not want to hide nodes or disable rules based on effectivity, do not specify an effective date or a Usage.

Whether you are creating a new configuration or restoring an existing configuration, the default effective date is the system date (that is, the current date and time of the database on which Configurator Developer is running).

For an overview of effectivity, see Chapter 6.

Enter a **Model Quantity** if you want to see how ordering more than one Model affects the configuration. This setting affects the unit testing session only if the Model is a BOM Model, or it is a non-imported Model that references a BOM Model.

To be able to modify the value of any Totals and Resources in the Model Debugger, select **Enable Editing of Totals and Resources**. Selecting this option provides another way to test rules that use Totals and Resources as participants.

22.2 The Model Debugger

The Model Debugger provides a view of the Model's hierarchical structure and allows you to test the structure and rules by selecting options, entering values, and adding component instances. You can also view a summary of your selections, save configurations, and restore saved configurations for additional testing.

The main benefit of using the Model Debugger is that a UI is not required to unit test a configuration model. The Model Debugger displays the entire Model structure, including non-BOM nodes, and provides different views of Model data based on your selections. You can also choose to test only specific areas of the Model, which may be time consuming in a UI if the Model is very large or if the UI is complex.

The Model Debugger displays each option's logic state, indicates whether a component contains required selections, and enables you to run any associated Configurator Extensions. The Model Debugger runs in the same browser window as Configurator Developer.

Like a UI generated in Configurator Developer, the Model Debugger uses images to indicate the selection state and status of each option, and displays an "icon legend" on each page. However, unlike a generated UI, you cannot control which images are used in the Model Debugger. The images that the Model Debugger uses are shown in Figure 20–2, "Default Selection State and Status Indicator Images" on page 20-10.

To access the Model Debugger, you must log into Oracle Applications and select one of the predefined Oracle Configurator Developer responsibilities. For a description of these responsibilities, see the *Oracle Configurator Implementation Guide*.

For more information, see Section 32.1, "Unit Testing Using the Model Debugger" on page 32-1.

22.3 Unit Testing a Generated User Interface

When unit testing a generated User Interface, verify that the UI has the look and feel that you want for your runtime Oracle Configurator and the screens present appropriate information to the end user in an easily usable format.

View and test the structure of your Model and confirm that only the nodes and Properties you want the end user to see are visible. For details, see Section A.3, "Configuring an Item in a Runtime Oracle Configurator" on page A-3.

As you test the Model, you may notice parts of the UI that you want to change or rules that do not function as you intended. Return to Configurator Developer to make any necessary changes. Before re-testing, be sure to regenerate logic and refresh the UI.

For more information on how Model structure and the User Interfaces you create in Configurator Developer are related, see Section 19.1 on page 19-1.

If you want to display pricing or Available to Promise (ATP) information in the UI, see Section 22.4 on page 22-3.

For more information, see Section 32.2, "Unit Testing Using a Generated User Interface" on page 32-4.

Note: Like other OA Framework-based applications, there is only one cache per Configurator Developer session. Therefore, if another user modifies a Template Reference in the UI that you are unit testing, the change appears at runtime when an action causes the page to be refreshed. For example, an Option Feature displayed as a group of radio buttons may change to a drop-down list at runtime when a Configurator Developer user changes the node’s Template Reference. When this occurs, Oracle Configurator does not display a message notifying the end user of the change.

22.4 Displaying Pricing Information and ATP Dates when Unit Testing

Pricing information is available only for items that exist in the CZ schema’s Item Master (see Chapter 2, "The CZ Schema’s Item Master"). Depending on your implementation, this information may include list prices, selling prices, and the extended price. (The extended price is the quantity multiplied by the selling price.)

If you want to display pricing or Available to Promise (ATP) information for items when unit testing using the Model Debugger or a runtime UI, perform the following:

- Enable pricing and ATP by defining the system property `cz.activemodel`. For details, see the *Oracle Configurator Installation Guide*.
- In the Preferences page, specify the Oracle Configurator pricing and ATP callback interface packages and procedures to use.

  For details, see Section 24.3.3.4, "Test Preferences" on page 24-10.

After specifying the pricing and ATP callback interface packages and procedures, note the following before unit testing:

- Before unit testing in a runtime User Interface, specify whether pricing and ATP information appears and what actions cause prices to be updated.

  For more information, see Section 31.3.1, "Modifying the User Interface Definition" on page 31-3.

- The Model Debugger updates both list prices and selling prices whenever you make a selection or navigate to another page. There are no settings in Configurator Developer that allow you to modify this behavior.

For additional information about pricing and ATP, see the *Oracle Configurator Implementation Guide*.

23 Publishing

This chapter provides information about publishing, which is typically the final phase of the configuration model development process.

This chapter includes the following sections:

- The Publishing Process
- Republishing
- Applicability Parameters
- Overlapping Applicability Parameters

23.1 Introduction

Creating configuration models is an iterative process in which you create a Model, test and update it, and then retest it until the Model is approved for production use. Typically, a configuration model is tested under a variety of conditions to prepare it for the various ways in which it will be used by Oracle Configurator end users to configure products and services. When a configuration model is ready for testing or production use, you must publish it. Publishing makes the Model structure, rules, and UI available to host applications such as Oracle Order Management or *i*Store.

Publishing is a two-step process. You first create a Model publication in Configurator Developer. Then, an Oracle Applications concurrent program copies all configuration model data to the database you specified when creating the publication. The target database can be the same database on which Configurator Developer is running, or a different one. The result of the copied configuration model data is called a publication.

A configuration model can have multiple User Interfaces and you can create many publications for the same Model. However, a publication corresponds to only one configuration model and User Interface.

When an Oracle Applications end user launches Oracle Configurator from a host application, the Configurator searches the database for the publication whose definition matches the information sent by the host application. If no matching publication is found but the Model was created from an imported BOM Model, Oracle Configurator displays the BOM Model in the Generic Configurator UI. If no matching publication is found and the Model was created in Configurator Developer, Oracle Configurator displays an error.

For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

All UI Content Templates that a UI references are published automatically when you publish (or republish) a Model.

23.2 Before Publishing

Before you publish a configuration model, be sure that you understand the publication process and carefully plan for how publications will be used.

Refer to the *Oracle Configurator Implementation Guide* for:

- Things to consider when planning publications
- Details about how a host application selects a publication
- A description of the database tables used during the publishing process
- Examples of how to maintain publications once they are available in your production environment For an example of how a sample organization maintains its Model publications, see the *Oracle Configurator Implementation Guide*
- An example of how a sample organization maintains its Model publications

23.3 The Publishing Process

Following is an overview of the publishing process:

1. Using Oracle Configurator Developer, create a publication request. This creates a new record in the database on which Configurator Developer is running.

   For details, see Section 27.2 on page 27-2.

2. Create the publication on the target database by running a concurrent program.

   For details, see Section 27.5 on page 27-5.

> **Note:** The publication cannot be viewed in a runtime Oracle Configurator until the publication concurrent program completes successfully.

23.3.1 Testing a Publication

Following is an overview of how to system test a publication:

1. Test the publication by invoking the Model from a host application such as Oracle Order Management, *i*Store, or TeleSales to ensure that it functions as intended.

   Refer to Chapter 22, and your host application's documentation, for more information.

2. Use Oracle Configurator Developer to make any necessary changes to the Model, rules, UI, or the publication's definition.

3. Republish the Model so the changes you made in Configurator Developer are visible to end users.

   Republishing is explained in Section 27.4 on page 27-5.

> **Note:** It may also be necessary to synchronize publication data for configuration models that are based on imported BOM Models. For more information about data synchronization, see the *Oracle Configurator Implementation Guide*.

4. When testing is complete, make the Model available for production use.

If you use the same database for both development and production activities, you can do this by changing the existing publication's Mode applicability parameter from Test to Production.

If you maintain separate development and production environments, create a new publication in your production database. If your system is set up correctly, you can do this by selecting your production database from a list.

23.4 Republishing

When Model data changes in Oracle Bills of Material, or you modify the Model structure, any rules, or its UI in Configurator Developer, you must **republish** the configuration model. Republishing updates an existing publication by copying all new and modified data to the target database. Republishing a Model and UI also updates any UI Content Templates that the UI references, since the templates may have changed since the Model was published.

You cannot modify a publication's applicability parameters when republishing. To modify a publication's applicability parameters, see Section 27.6 on page 27-6.

When you republish a Model, Configurator Developer creates a new publication record, adds a new publication record ID at the end of the existing publication, and changes its status to Pending Update. The new publication ID that is appended to the existing record indicates that the two records are related.

For example, publication record 1001 exists for Model M1 and its status is Complete. Table 23–1 shows how Configurator Developer creates a new record (ID # 1002) when you create a request to republish Model M1.

Table 23–1 Republishing a Model

ID	Model	UI	Published	Status
1001	M1	UI-1	11/23/2000 12:01	Pending Update {1002}
1002	M1	UI-1	11/25/2000 14:35	Pending

If you decide that you do not want to create the publication, you can delete the Pending process (ID 1002) before its status changes to Complete. Deleting a publication is explained in Section 27.7 on page 27-7.

Table 23–2 shows how the new publication record's status changes to Processing when one of the publishing concurrent program selects it.

Table 23–2 Processing a Republish Concurrent Request

ID	Model	UI	Published	Status
1001	M1	UI-1	11/23/2000 12:01	Pending Update {1002}
1002	M1	UI-1	11/25/2000 14:35	Processing

Table 23–3 shows how Configurator Developer updates the publication records when the concurrent program successfully copies the Model data to the target database, and creates the publication on the target database. That is, the status of the new record changes to Complete and the old record (ID 1001) no longer appears.

Table 23–3 Updating Status when Republish is Successful

ID	Model	UI	Published	Status
1002	M1	UI-1	11/25/2000 14:35	Complete

Table 23–4 shows how Configurator Developer updates the publication records when the concurrent program fails. In this case, the status of the original publication reverts to Complete and the new record is set to Error.

Table 23–4 Updating Status when Republish Fails

ID	Model	UI	Published	Status
1001	M1	UI-1	11/23/2000 12:01	Complete
1002	M1	UI-1	11/25/2000 14:35	Error

When a publishing request fails, you must manually delete the publication request (in this example, ID 1002). For details, see Section 27.7 on page 27-7.

23.5 Applicability Parameters

You specify applicability parameters to control a publication's availability to host applications. When an end user invokes Oracle Configurator, the host application creates an **initialization message.** To display the publication, the parameters in this message must exactly match the publication's applicability parameters; otherwise, the runtime Oracle Configurator displays an error. For more information about the initialization message, see the *Oracle Configurator Implementation Guide*.

You can create multiple publications for the same configuration model on the same database, but every publication's applicability parameters must be unique. In other words, publications with *overlapping* applicability parameters cannot exist on the same database. For details, see Section 23.6 on page 23-5.

Applicability parameters determine which publication to display based on the following criteria:

- **Mode**: Use this parameter to specify whether the publication is used for production or testing activities, or if it is unavailable. Values include Test, Production, and Disabled. Only publications with a mode of Production or Test can be accessed by a host application such as Order Management or *i*Store. A publication marked as Disabled is unavailable to host applications, but is *not* deleted from the database.
- **Applications**: Use this parameter to specify which host applications can access the publication. For example, you can specify that a publication is available to Oracle Order Management and *i*Store, but not TeleSales. For more information about this parameter, see the chapter on publishing in the *Oracle Configurator Implementation Guide*.

  For a complete list of host applications, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.
- **Languages**: Use this parameter to specify in which languages the publication is available. You must select at least one language. The list of values includes the base language and all installed languages on the Oracle Applications database on

which Configurator Developer is running. For more information, see Appendix B, "Multiple Language Support".

- **Usages**: Use this parameter to indicate whether the publication is available based on the Usage specified by the host application. By default, a publication's availability is not limited to a specific Usage, and the Usages applicability parameter is set to "Any Usage."

  Oracle Applications products (such as Order Management and iStore) use a profile option to determine the Usage name to use when selecting a publication. For details, see the *Oracle Configurator Installation Guide*.

  For general information about Usages, see Chapter 6.

  For an example of how you can use Usages to limit a publication's availability, see the *Oracle Configurator Implementation Guide*.

- **Date Range**: Use this parameter to indicate that the publication is always available, never available, or is available only for a specific period of time. The Start and End dates may range from 01/01/1601 through 12/31/4710, inclusive.

  The dates and times you specify here determine only whether the *publication* is available when accessed by a host application; the effective dates (or Effectivity Set) defined within the Model itself determine whether parts of the Model structure, configuration rules, or UIs are available at runtime.

> **Note:** Oracle Configurator compares the time period you specify for the Date Range parameter to the system date of the target database in which the publication exists. If the time on the machine on which the publication was created and the time on the machine on which the host application is running are not the same, the Model may not appear as expected. To resolve this issue, override the default **Valid From** setting by selecting the **No Start Date** box.

23.6 Overlapping Applicability Parameters

Only one publication can exist on the same database for the same product, publication mode, application, Usage, and so on. If other publications currently exist on the target database, at least one applicability parameter must be unique to create the new publication. Oracle Configurator Developer ensures that any changes you make to existing publications will not create conflicting applicability parameters.

Table 23–5 provides examples of how Configurator Developer does not allow multiple publications for the same Model on the same target database instance.

Table 23–5 Comparing Applicability Parameters

Applicability Parameter	Publication A	Publication B	Publication C	Publication D
Publication Mode	Test	Test	Test	Test
Applications	OM	*i*Store, Order Capture, OM	OM	OM
Usage	Desktop_A1	Desktop_A1	Desktop_A1	LaptopPC
Date Range	10-JUN-00 to 01-DEC-00	10-JUN-00 to 01-DEC-00	10-OCT-00 to 10-DEC-00	10-JUN-00 to 01-DEC-00
	Creation Status: Successful	Creation Status: Failed	Creation Status: Failed	Creation Status: Successful

In this example, the configuration model is successfully published and the result is Publication A and Publication D. When you compare this publication with requests to create publications B and C, you can see that:

- The request to create Publication B fails because although the publication is available to *i*Store and Order Capture, it is also available to Order Management (OM). Therefore, the Applications parameters overlap.
- The request to create Publication C also fails because the date ranges overlap.

Publication D is created successfully because the Usage specified is unique, even though all of its other applicability parameters are the same as Publication A's.

For more information about maintaining publications, see the *Oracle Configurator Implementation Guide*.

Part VI

Developer Tool Reference

Part VI describes the Configurator Developer user interface and presents step-by-step instructions for performing required configuration model development tasks.

Part VI contains the following chapters:

24 Configurator Developer User Interface Basics

This chapter provides general information about the Configurator Developer user interface.

This chapter includes the following sections:

24.1 The Configurator Developer User Interface

The Configurator Developer user interface contains two main tabs that are labeled Repository and Workbench.

The Repository tab contains the following areas: Main, Item Master, and Publishing. You use these areas to organize Models and shared objects in a single location. Shared objects include Usages, Effectivity Sets, Properties, Items in the CZ schema's Item Master, Configurator Extension Archives, and User Interface Templates. You use the Publications area of the Repository to publish Models for testing and production activities.

The Workbench tab contains the following areas: General, Structure, Rules, and User Interface. You use these areas to create and maintain Model structure, rules, and User Interface(s). Data in these areas appear in a hierarchical structure. For details, see Section 1.1.4, "Hierarchical Structure" on page 1-2.

In each area of the Repository and the Workbench, each object's name appears as a hypertext link. When you click the link,Configurator Developer displays the object's read-only details page. If the object can be modified, you can click an Edit button to open it for editing. You can also click the icon in the Edit column to open an object for editing directly from each area of the Repository or Workbench.

For an overview of using Oracle Configurator Developer tools to build a configuration model, see Section 1.3.4, "Build a Configuration Model" on page 1-5.

Many elements of the Configurator Developer user interface are common to other OA Framework-based applications. For more information about common UI elements, see the *Oracle Applications User's Guide*.

24.1.1 Views

You can customize the appearance of most Repository and Workbench areas using Views. Views control the order and title of table columns, and whether the columns appear when the View is applied. A list of available Views appears in the Main, Item Master, and Publications areas of the Repository, and the Structure, Rules, and User Interface areas of the Workbench. You apply a View by selecting it from the View list and then clicking Go.

Configurator Developer provides several predefined Views for areas that contain hierarchical tables. These Views are available for all users and can be set based on security level by the Configurator Administrator (in other words, a specific view may be assigned for each role). The predefined Views cannot be updated or deleted, but you can copy a predefined View and modify it, or create your own Views. You can also specify whether a View is used by default when you start Configurator Developer.

For more information, see the *Oracle Applications User's Guide*.

To create a new View:

1. Click **Personalize**.
2. Click **Create View**, or select an existing View and then click **Duplicate**.
3. Enter a **View Name** and optionally a **Description**.
4. To apply this View whenever you begin using Configurator Developer, select **Set as Default**.
5. From the list of **Available Columns**, specify which columns appear when the View is applied.

   To add a column, either double click the column name, or select the column, and then click **Move**. The column(s) you selected appear in the **Columns Displayed** list.

   Click **Move All** or **Remove All** to add or remove all columns in a list.
6. Specify the order in which you want the columns to appear by selecting the column name, and then clicking the up or down arrows within the **Columns Displayed** list.
7. To rename any column headings, click the **Rename Columns/Totaling** button.

   In the Rename Columns/Totaling page, modify column names as necessary, then click **Apply**.
8. Click **Apply**.
9. Click the link at the bottom of the Views page to return to the Repository or Workbench area in which you were working.

To modify a View:

1. Click **Personalize**.
2. Select a View, then click **Update**.
3. Modify the View as required, then click **Apply**.

To delete a View:

1. Click **Personalize**.
2. Select a View, then click **Delete**.
3. Click **Yes** to confirm the action.

24.1.2 Search

Some Models may have a large number of structure nodes or many rules, and it can be difficult to locate the one you want to select, view or edit. To help you locate objects, a **Simple Search** button appears in many Configurator Developer pages. For example, you can search for a Model in the Main area of the Repository, an Item in the Item Master area of the Workbench, or Model structure nodes when defining a rule. You can also perform an advanced search, which enables you to enter additional search criteria, and save a search as a View. See Section 24.1.2.1 on page 24-3.

When searching for objects, you can also use **wild-card characters**. The wild-card for a complete string is "%" (the percent sign). The wild card for a single character is the underscore character (_). If you need to search for an object whose name contains one of the wild-card characters, prefix the wild-card character with the escape character (/).

Table 24–1 provides examples of search results using wild-card characters.

Table 24–1 Searching for Model Nodes Using Wildcard Characters

Search Criterion	Finds
Feature/_10%	Feature_1040, Feature_1041, Feature_10 abc
Feature/_10_	Feature_104, Feature_10A

You can search for objects using the following criteria:

- Name
- Description
- Type or Node Type: For example, when searching for objects in the Main area of the Repository, enter Model, Folder, Property, or UI Content Template. When searching in the Structure area of the Workbench, enter BOM Option Class, Component, or Boolean Feature. In the Rules area of the Workbench, enter Logic Rule, Rule Folder, or Statement Rule.
- Locked: For example, enter Yes to view all locked objects, or No to view all unlocked objects.
- Locked By: For example, enter your user name to view all objects that are currently locked by you.

For more information about searching for objects in Configurator Developer, see the *Oracle Applications User's Guide*.

24.1.2.1 Saving a Search as a View

You can save searches that you use often as a View. This is useful if you want to see only a specific set of objects at a time when working in Configurator Developer.

For example, you want to be able to quickly locate and begin modifying a specific Model each time you log into Configurator Developer. You search for the Model and save the search as your default View. The next time you log into Configurator Developer, the Model will be the only one that appears in the Main area of the Repository. If you do not set it as your default View, you can apply it at any time by selecting it from the View list and clicking **Go**.

Views are described in Section 24.1.1 on page 24-2.

To save a search as a View:

1. Click **Simple Search**.

2. Enter search criteria, and then click **Go**.
3. Click **Save Search**.
4. In the Create View page, enter a **View Name** and optionally a **Description**.
5. Optionally modify the list of columns that appear in the View, whether to make this View your default, and so on.
6. Click **Apply** or **Apply and View Results**.

24.1.3 Actions

Use the **Actions** list to modify Configurator Developer objects. The list of available actions is determined by the functions included in your login responsibility and the area of the Repository or Workbench in which you are currently working. The list of actions may include Copy, Delete, Move, Rename, Reorder Children, and so on.

To perform an action:

1. Select one or more objects.
2. Choose an option from the **Actions** list, then click **Go**.
3. Enter additional information as required, then click **Finish**. For example, if you are copying a Model, you must specify the Folder in which you want to store the copy.

For more information, see:

- Section 25.5, "The Main Area of the Repository Actions List" on page 25-9
- Section 26.8, "The Item Master Area of the Repository Actions List" on page 26-3
- Section 29.18, "The Structure Area of the Workbench Actions List" on page 29-15
- Section 30.14, "The Rules Area of the Workbench Actions List" on page 30-16
- Section 31.3.2, "Copying, Moving, and Deleting User Interface Elements" on page 31-5

For more information, see the *Oracle Applications User's Guide*.

24.1.4 Shortcut Links

Shortcut links appear in any page that has multiple, clearly defined sections. These links eliminate the need for scrolling when all of a page's content will not fit within the visible area. For example, a Model node's details page may have several sections, such as Properties, Effectivity, Associated Rules, and so on. Instead of scrolling, you can click on the Associated Rules shortcut link to "jump" to that section of the page. Each target section includes a Return to Top link, which returns you to the top of the current page.

For more information, see the *Oracle Applications User's Guide*.

24.1.5 The Focus Column

Clicking the icon in the Focus column allows you to "focus in" on a specific object in a hierarchical table. When you do this, Configurator Developer displays the object as the "root" of the hierarchical table and expands its contents to one level so all of its children are also visible. The Focus icon is useful when a hierarchical table contains many objects because it enables you to locate the object you want to view or open for editing more easily.

For example, when you click the icon in the Focus column next to a Folder:

- The Folder becomes the root of the table you are viewing
- All of the Folder's child objects are visible
- The other objects that were at the same level as the Folder no longer appear

The Focus icon is available only for parent or container objects. In other words, you cannot focus on objects that do not have children. Parent objects include Model structure nodes and some UI elements (such as UI Pages and Layout Regions). Container objects include Folders and the root node of a UI

After focusing on a parent or container object, you can use the locator links at the top of the table to return to the previous display and view all objects in the table.

24.1.6 Printable Pages

A Printable Page button appears in many read-only pages in Configurator Developer. For example, this button appears when you open a Model for viewing by clicking on its name in the Main area of the Repository. Click this button to display the page in a separate window in a format that is more suitable for printing. You can then send the page to a local printer using your browser's Print option.

24.2 Locking Models and UI Content Templates

Because multiple users may be using Configurator Developer simultaneously, by default you must lock Models and UI Content Templates before editing them. This prevents other users from modifying the object at the same time. You or the Configurator Developer Administrator must unlock a Model or UI Content Template before another user can modify it.

Locking a Model does not lock any of its referenced Models.

Locking is controlled by a Site-level profile option, which means it affects all Oracle Configurator Developer users on a specific instance. The System Administrator can change the default behavior by modifying this profile option, but this is not recommended. For details, see the *Oracle Configurator Installation Guide*.

WARNING: Configurator Developer refers to each user's Oracle Applications user name when locking and unlocking objects. Therefore, it is essential that all Configurator Developer users have a unique Oracle Applications user name and password when locking is enabled. In other words, multiple users should not be able to log into Oracle Applications with the same user name and password, and then work in Configurator Developer.

Defining Oracle Applications users is described in the *Oracle Applications System Administrator's Guide*.

By default, locking is required to:

- Edit a Model (includes creating or modifying Model structure, defining or modifying rules, and generating or editing User Interfaces)
- Refresh or publish a Model, refresh a Model's UIs, or generate logic

  For details, see Section 24.2.2, "Automatic Model Locking" on page 24-6.
- Edit a UI Content Template

If an object is currently locked by another user, the icon in the Edit column is disabled. However, you always have read-only access to a locked object and can view its details. You can also view a Model or UI Content Template's details without locking it.

You do not have to lock a Model or UI Content Template before copying or deleting it, but it cannot be locked by another user.

Additionally, if a Model has one or more referenced Models that are locked by another user, you cannot:

- Copy the Model
- Delete the Model
- Generate logic for the Model
- Refresh any of the Model's UIs

In each case, a message indicates which Models are locked, and by whom.

You do not have to lock an object to move or rename it, and you can perform these actions even if the object is locked by another user. You can also generate a Model Report for a Model that is locked by another user.

24.2.1 Force Unlock

If you are logged in as the Oracle Configurator Administrator, you can unlock any object. In other words, you can unlock an object that is locked by another user. However, this should be done only when absolutely necessary, such as when a user forgets to unlock a Model and then is unavailable for a period of time. Configurator Developer displays a confirmation message when you attempt to unlock an object that is locked by another user.

If you have a Model or UI Content Template open for editing and the Oracle Configurator Administrator unlocks it, Configurator Developer displays an error message when you attempt to modify an object. For example, you are editing a Model and are viewing the Structure area of the Workbench when the Oracle Configurator Administrator unlocks the Model. When you try to edit a Model node (by clicking the icon in the Edit column), an error message appears in the node's details page indicating that the Model is no longer locked by you and you cannot make any changes. When you close the node's details page and return to the Structure area of the Workbench, you have read-only access to the Model. You must return to the Main area of the Repository and lock the Model before you can edit it.

If you are modifying a Model or UI Content Template and are viewing an object's details page when the Oracle Configurator Administrator unlocks the object, you can save any pending changes. However, you cannot make any additional changes because the object is no longer locked.

For example, you have a UI Content Template open for editing and are viewing a UI element's details page when the Oracle Configurator Administrator unlocks the template. When you click Apply, Configurator Developer saves your changes but displays a message indicating that the template is no longer locked and you now have read-only access to it. You must return to the Main area of the Repository and lock the template before you can edit it.

24.2.2 Automatic Model Locking

When you submit an Oracle Applications concurrent request to import or refresh a BOM Model, the Model and all of its referenced Models must either be unlocked or locked by you.

A similar requirement applies when publishing a Model, generating logic, or refreshing UIs. When you submit an Oracle Applications concurrent request to publish Model data, the Model, any referenced Models, and any referenced UI Content Templates must either be unlocked or locked by you. This requirement does not apply when you are creating a publication in Configurator Developer. This procedure is described in Section 27.2, "Creating a New Model Publication" on page 27-2.

If a Model references other Models, all of the referenced Models and referenced UI Content Templates must either be unlocked or locked by you before generating logic or refreshing the Model's UI(s).

If none of the referenced objects are currently locked when you perform one of these tasks, Configurator Developer locks them until the operation is complete. Configurator Developer then unlocks all of the referenced objects.

If any of the referenced objects are locked by another user, an error message lists which objects are currently locked and by whom. In this case, you must either ask the other user(s) or the Oracle Configurator Administrator to unlock the object(s) before you can continue.

For general information about publishing, see Chapter 23, "Publishing".

For details about the concurrent programs referred to in this section, see the *Oracle Configurator Implementation Guide*.

24.2.3 Locking an Object and Viewing Locked Status

Following are the locking-related columns that may appear in the Main area of the Repository when locking is enabled:

- **Locking**: Use the icon in this column to lock or unlock a Model or UI Content Template. If you locked the object, the icon is enabled and you can use it to unlock the object. If another user locked the object, the icon is disabled.

  An icon does not appear in this column for predefined UI Content Templates, because these templates cannot be modified.

  The icon is always enabled if you logged in as the Oracle Configurator Administrator. See Section 24.2.1, "Force Unlock" on page 24-6.

  The Locking column does not appear if your responsibility does not allow you to modify Models or UI Content Templates. For example, the Locking column does not appear when you select the Oracle Configurator Viewer responsibility.

- **Locked**: A `Yes` or `No` in this column indicates whether the object is currently locked.
- **Locked By**: This column displays the name of the user who locked the object.
- **Locked Date/Time**: This column indicates when the object was locked.

When locking is enabled, you can control whether any of these columns appear by modifying the current View. These columns are not available when locking is disabled (that is, when the profile option CZ: Require Locking is set to `No`). For details, see Section 24.1.1, "Views" on page 24-2.

You can also perform a search to view a list of all locked or unlocked objects. For details, see Section 24.1.2, "Search" on page 24-3.

24.3 Global Links

Global links appear in every page in the Configurator Developer user interface. These links display a separate page in which you can view information or perform additional actions.

For more information about global links, see the *Oracle Applications User's Guide*.

24.3.1 Home

Click this link to close Configurator Developer and return to the Oracle Applications E-Business Suite Home page.When you do this, Configurator Developer prompts you to save any pending changes.

24.3.2 Logout

Click this link to close Configurator Developer and log out of Oracle Applications. When you do this, Configurator Developer prompts you to save any pending changes.

24.3.3 Preferences

Click the Preferences global link to display the Preferences page. The settings in this page enable you to define user-specific settings that apply when you are working in Oracle Applications and Configurator Developer. When you navigate to the Preferences page from the E-Business Suite Home page, you can view and modify user-specific settings that affect all of the Oracle Applications products to which you have access. These settings include your default language, time zone, notification preferences, and your Oracle Applications user name and password. These settings are described in detail in the *Oracle Applications User's Guide*.

When you navigate to the Preferences page from Configurator Developer, you can also define settings that affect the display of Model structure nodes and tables in Configurator Developer. These settings affect all Models and pages in the CZ schema to which you are connected. Additionally, these settings are user-specific, so they remain in effect even after you exit and then restart Configurator Developer.

You can also enter custom initialization parameters used by the runtime UI and the Model Debugger when unit testing the configuration model. These settings are described below.

24.3.3.1 Display Preferences

The Number of Rows Shown in Tables setting controls how many rows are displayed in tables. For example, this setting affects tables that appear in a node's details page (in the Associated Properties, Associated Rules, and Associated UI Nodes sections), as well as the Add to Selected List table, which appears in the Choose Nodes page.

If a table contains more rows than the number you specify here, Configurator Developer provides links above and beneath the table so you can view the additional rows.

This setting does not affect how many rows appear in pages that display data in a hierarchy, such as all areas of the Repository and Workbench. This value is determined by the profile option CZ: Number of Table Rows Displayed. For details, see the *Oracle Configurator Installation Guide*.

24.3.3.2 Effectivity Date Filter

Use this setting to control whether Model structure nodes and rules that are not effective appear when working in Configurator Developer. For general information about effectivity, see Chapter 6, "Effectivity".

Select one of the following options:

- Select **All** if you do not want to hide any Model nodes or rules based on their effectivity. This is the default value.
- Select **Current** to display only Model nodes and rules that are effective "now" (that is, as of the current date and time). In other words, the date and time when you are working in Configurator Developer must fall between the object's effective Start Date and End Date. (Start Date <= Current Date < End Date)
- Select **Future and Current** to display only Model nodes and rules that are effective now or in the future, regardless of their start date. (End Date > Current Date)

The Effectivity Date Filter setting affects all Models you can view or modify in Configurator Developer, and it considers both effective date ranges and Effectivity Sets. It does not consider Usages. This setting is stored as a user-level profile option and persists across Configurator Developer sessions. In other words, it does not change when you exit and then log back into Configurator Developer.

Model nodes whose effectivity does not match the Effectivity Date Filter setting do not appear in the Structure area of the Workbench. Similarly, any rules whose effectivity does not match this setting do not appear in the Rules area of the Workbench. In the UI area of the Workbench, nodes that are not effective do not appear when you are specifying a UI element's Associated Model Node or defining a runtime condition.

For example, if the Effectivity Date Filter is set to Current and the current date is 22-Feb-05, any nodes or rules whose effectivity does not include this date do not appear.

Configurator Developer also considers this setting when:

- You are defining a rule or modifying a UI element and must select a node from the Model structure
- You are viewing a Model node's details page (in other words, any rules that are not effective do not appear in the Associated Rules section)
- You use the Search feature to locate a specific node(s) or rule(s)

Configurator Developer does *not* consider this setting when you are viewing:

- A rule's details page: The Model nodes that participate in a rule are always visible here, regardless of the Effectivity Date Filter setting.
- A User Interface's structure: When the Model Node column is displayed in the UI area of the Workbench, a UI element's associated Model node always appears in this column, regardless of its effectivity.
- A UI element's details page: The element's associated Model node is always visible here, regardless of the Effectivity Date Filter setting.
- A list of Effectivity Set members: See Section 25.5.5, "List Effectivity Set Members" on page 25-12.

Model nodes and rules that are set to Always Effective are always visible, regardless of the Effectivity Date Filter setting.

If a Model node is not displayed due to this setting, none of its descendants are visible either (regardless of their effectivity).

24.3.3.2.1 Effectivity Date Filter Setting Display The current value of this setting appears at the top of each Configurator Developer page for reference. It also appears at the top of the Associated Rules section in all Model node details pages.

For example:

`Date Effectivity:` **`Future and Current`**

24.3.3.3 Structure Node Display

The settings in this section control how Configurator Developer displays Model structure nodes in rules and in a UI element's details page.

Settings are available for controlling the display of both BOM and non-BOM nodes. The default values settings are:

- BOM Structure Nodes: View by Description
- Non-BOM Structure Nodes: View by Name

These settings do not affect how nodes are displayed in the Structure area of the Workbench. In the Structure area of the Workbench, node names always appear and you can display node descriptions by applying a View that contains the Description column. For more information, see Section 24.1.1, "Views" on page 24-2.

If a node does not have a description and either setting is set to View by Description, Configurator Developer displays the node's name instead. (The opposite is not true, since all nodes must have a name.)

When a node that is a participant in a rule is deleted, it appears as "DELETED: *Node Name or Description*" in the rule's details page.

If the Structure Node Display settings are different for BOM and non-BOM nodes, and a Statement Rule contains both types of nodes, both node names and descriptions appear when viewing the rule's definition in the Rule Statement for Display section. Statement Rules are described in Chapter 16.

Configurator Developer uses the Structure Node Display settings when constructing a node's path, which indicates its location in the Model. This information appears when viewing the details page for a Model structure node, rule, or UI element. For example, for your Model, BOM Structure Nodes is set to Description. When you open the details page for a BOM item, its path appears like this:

`Premium Custom Laptop.Hard Drive Option Class.40 GB Hard Drive`

If BOM Structure Nodes is set to Name, the same BOM item's path appears like this:

`CN62441C.OC68020.CM41020`

You can also view an object's path by placing the cursor over its associated icon. For example, when viewing a rule's details page, you can view the path of each node that is a participant in the rule.

24.3.3.4 Test Preferences

In the Custom Initialization Parameters field, enter any custom XML parameters you want to use when unit testing the selected Model in a runtime User Interface or the Model Debugger. For example, to display pricing and Available to Promise (ATP) information when unit testing, specify the required interface PL/SQL packages and procedures here.

This field accepts a maximum of 300 characters.

Enter initialization parameters using the following syntax:

```
<param name="parameter_name">parameter_value</param>
```

For example:

```
<param name="pricing_package_name">cz_price_test</param>
<param name="price_mult_items_proc">price_multiple_items</param>
<param name="price_single_item_proc">price_single_item</param>
<param name="atp_package_name">cz_atp_callback_stub</param>
<param name="get_atp_dates_proc">call_atp</param>
```

Note: Be sure the parameters you enter follow this syntax and do not contain any typographical or spelling errors. Configurator Developer does not validate the syntax, and an incorrectly formatted parameter will produce an error at runtime.

When launching the runtime UI or Model Debugger from Configurator Developer, the custom parameters that you enter here are prepended to the initialization message. In other words, any parameters that you enter in Configurator Developer when launching a UI or the Model Debugger take precedence over the parameters you enter in this page.

For example, if you enter an effective date as a custom initialization parameter, and then enter a different date as a session parameter before unit testing a configuration model, the date you entered as a session parameter is used. Unit testing is described in Chapter 22.

When unit testing a configuration model, you cannot display pricing or ATP data by specifying the procedures and packages used by Oracle Applications products, such as Oracle Order Management. Specify parameters for displaying pricing and ATP data only if you want to test your own custom packages and procedures. For details about pricing and ATP when launching Oracle Configurator from a host application, see the *Oracle Configurator Implementation Guide*.

24.3.4 Help

Click this link to display the Oracle Applications online help system. This system includes the entire contents of the *Oracle Configurator Developer User's Guide* and the *Oracle Configurator Constraint Definition Language Guide*.

The online help is context-sensitive. This means that if you are working in the Rules area of the Workbench, for example, clicking the Help link displays information about creating rules.

24.3.5 Diagnostics

Click this link to select what type of logged messages to display while working in Configurator Developer. The options available in the Diagnostics page include Show Log, Set Trace Level, and Show Log On Screen.

For more information about these options, see the Oracle Applications Framework Release 11i Documentation Road Map (Metalink Note # 275880.1).

24.4 Navigation and Saving Data

When you navigate to another page before saving, Configurator Developer prompts you to either save your work, or cancel the pending transaction. For example, you are editing a rule and make some changes. If you click the Structure link to navigate to that area of the Workbench, Configurator Developer indicates that changes are pending and you must choose how to proceed. All changes you make in Configurator Developer are stored directly in the CZ schema when you click Apply or Finish at the end of a process flow. There is no "undo" feature, so if you create an object and then decide you do not need it, you must manually delete it.

WARNING: Do not delete objects from the CZ schema directly using PL/SQL or SQL*Plus. The concurrent program Purge Configurator Tables is the only supported method of removing logically-deleted records in the CZ schema.

All Configurator Developer pages provide navigation controls such as buttons that enable you to apply or cancel a pending transaction. It is strongly recommended that when using Configurator Developer, you always use the buttons and links provided in the Configurator Developer user interface and not your Web browser's navigation controls. Using the browser's Back, Next, or Home buttons, for example, can result in errors or even loss of data.

However, you can use the browser's Back button when an unrecoverable error occurs. In this case, no navigation controls appear in the Configurator Developer user interface, and the Back button is the only way to return to the page you were viewing when the error occurred.

24.4.1 Keyboard Shortcuts

Configurator Developer is an HTML-based application that provides various keyboard shortcuts to cut, copy, and paste text in input fields, select check boxes and radio buttons, and use the arrow keys to select options from a drop-down list. You navigate to each control on a page from left to right, and from the top of the page to the bottom, by pressing the Tab key. Press Shift+Tab to navigate in the opposite direction. Press Enter to execute the currently selected control, such as a button.

Table 24–2 lists the keyboard shortcuts available in Configurator Developer.

Table 24–2 Configurator Developer Keyboard Shortcuts

Command	Shortcut
Cut	Ctrl-x
Copy	Ctrl-c
Paste	Ctrl-v
Activate UI control	Tab
Execute active UI control	Enter

25 Main Area of the Repository

The Main area of the Repository appears when you begin using Configurator Developer. This area displays objects for organizing and defining configuration models in a hierarchical table.

This chapter presents information about the Main area of the Repository, including the following topics:

- The Main Area of the Repository Hierachical Table
- Main Area of the Repository Objects
- Creating Objects in the Main Area of the Repository
- Opening Objects For Editing
- The Main Area of the Repository Actions List

See the *Oracle Application User's Guide* for information about other parts of the Main area of the Repository that are standard elements of an HTML-based Oracle Applications user interface.

See Chapter 24, "Configurator Developer User Interface Basics" for help with elements that are generally available in Configurator Developer pages.

25.1 The Main Area of the Repository Hierachical Table

By default, all objects in the Main area of the Repository are displayed in a hierarchical table. All Folders appear first (that is, at the top of the page), in alphanumeric order. All other objects then appear in alphanumeric order after the last Folder. Folders are containers that may contain other Folders, as well as Models, Effectivity Sets, Usages, User Interface Templates, Configurator Extension Archives, and User Properties. All Folders are collapsed when you start Configurator Developer, but you can expand or focus in on a specific Folder to view its contents, and then select the object you want to modify.

A predefined Folder called Repository Main appears at the top of the table. You can create objects in this Folder, but the Folder's name and description are read-only.

The name of each object in the Main Repository is a link. Clicking an object's name displays its details in a read-only page. You can then edit the object by clicking an Edit button. To open the object directly from the Main area of the Repository, click the icon in the Edit column.

The Main area of the Repository also contains a Folder named UI Templates. This Folder stores all of the predefined UI Master Templates and UI Content Templates. For more information, see Chapter 20, "User Interface Templates".

25.2 Main Area of the Repository Objects

The following objects appear in the Main area of the Repository:

- Folders
- Models
- Effectivity Sets (see Chapter 6)
- Usages (see Chapter 6)
- Properties (see Chapter 5)
- Configurator Extension Archives (see Chapter 17)
- User Interface Master Templates (see Chapter 20)
- User Interface Content Templates (see Chapter 20)

25.2.1 Folders

Use Folders to store and organize objects in the Main area of the Repository. For example, you may want to create your own Folder to store all Models that you create or are currently working on, or group all of the UI Content Templates that you create in a specific Folder. A Folder can contain any object, including other Folders.

Folders must have a name. The root Folder's name is read-only.

To create a Folder, see Section 25.3.2 on page 25-3.

25.2.2 Models

The Main area of the Repository lists both BOM Model nodes and non-imported Model nodes. BOM Model nodes must be imported before they appear in the Main area of the Repository. See the *Oracle Configurator Implementation Guide* for information about importing BOM Models.

To create a Model, see Section 25.3.1 on page 25-3.

25.3 Creating Objects in the Main Area of the Repository

Following is the general procedure for creating objects in the Main area of the Repository:

1. Identify the row containing the intended parent of the object you are going to create.
2. In the parent object row, click the icon in the **Create** column.
3. From the list of available Object Types, select the desired object.

   For example, select **Folder**.
4. Click **Continue** and enter a **Name** and a **Description**.

   All objects of the same type must have a unique name.

   To create multiple objects of the same type, click **Add Another Row**, then enter a **Name** and **Description** for each.
5. Click **Finish**.

For specific examples, see the following sections:

- Creating a Folder

- Creating a Property
- Creating an Effectivity Set
- Creating a Usage
- Creating a Configurator Extension Archive
- Creating a User Interface Master Template
- Creating a User Interface Content Template (Chapter 31)

When you create a Model, Effectivity Set, Usage, or Property in the Main area of the Repository, it appears at the top level in the hierarchical table (that is, in the root Folder). You can then move the new object to a specific Folder. For details, see Section 25.5.1, "Moving and Copying Objects" on page 25-9.

You can create any type of object in the Main area of the Repository by copying an existing object of that type. For example, to create a UI Content Template, copy one of the predefined UI Content Templates then modify it to meet your needs.

25.3.1 Creating a Model

For general information about Models, see Section 3.2 on page 3-1.

To create a Model:

1. From the Main area of the Repository, in the same row as the Folder in which you want the Model to appear, click the icon in the **Create** column.
2. From the list of available Object Types, choose **Model**, and then click **Continue**.
3. Enter a **Name** and optionally a **Description**.

   Model names must be unique.
4. To create another Model, click **Add Another Row**.
5. Click **Finish**.

25.3.2 Creating a Folder

For general information about Folders, see Section 25.2.1 on page 25-2.

To create a Folder:

1. In the same row as the root Folder, or a user-created Folder, click the icon in the **Create** column.
2. Select an Object Type of **Folder**, then click **Continue**.
3. Enter a **Name** and, optionally, a **Description**.

   To create multiple Folders, click **Add Another Row**, then enter a **Name** and, optionally, a **Description** for each.
4. Click **Finish**.

25.3.3 Creating a Property

For general information about Properties, see Chapter 5.

To create a new Property:

1. In the row of the root Folder or a user-created Folder, click the icon in the **Create** column.

2. From the list of available Object Types, select **Property**, then click **Continue**.
3. Enter a unique **Name** and optionally a **Description**.
4. Select a **Data Type**, and enter a **Default Value**.

   For a description of each Data Type, see Section 5.6 on page 5-11.
5. Click **Apply**.

You add Properties to a Model node in the Structure area of the Workbench. For details, see Section 29.14 on page 29-9.

25.3.4 Creating an Effectivity Set

For general information about Effectivity Sets, see Section 6.3 on page 6-2.

There is no limit to how many Effectivity Sets you can create.

To create an Effectivity Set:

1. Click the icon in the **Create** column.

   If you want the Effectivity Set to appear in a Folder, click the icon in the same row as the Folder.
2. From the list of available Object Types, select **Effectivity Set**, then click **Continue**.
3. Enter a **Name** and **Description**, then click **Finish**.

See Section 25.4.3, "Modifying Effectivity Sets and Usages" on page 25-6.

25.3.5 Creating a Usage

For general information about Usages, see Section 6.4 on page 6-2.

To create a Usage:

1. Click the icon in the **Create** column.

   If you want the Usage to appear in a Folder, click the icon in the same row as the Folder.
2. From the list of available Object Types, select **Usage**, then click **Continue**.
3. Enter a **Name** and **Description**, then click **Finish**.

   Do not enter "Any Usage" as the Usage name. This name is reserved by Oracle Configurator Developer.

Note: You can create a maximum of 64 Usages.

25.3.6 Creating a Configurator Extension Archive

Before you can fully define a Configurator Extension, you must create a Configurator Extension Archive. See Chapter 17, "Configurator Extensions" and the *Oracle Configurator Extensions and Interface Object Developer's Guide* for background.

To create a Configurator Extension Archive:

1. Click the icon in the **Create** column.

   If you want the Configurator Extension Archive to appear in a Folder, click the icon in the same row as the Folder.

2. From the list of available Object Types, select **Configurator Extension Archive**, then click **Continue**.
3. Enter a **Name** and **Description**, then click **Finish**.

   The new Configurator Extension Archive appears in the Main area of the Repository, in the Folder where you created it.

   To complete the definition of a Configurator Extension Archive, see Section 25.4.4, "Modifying Configurator Extension Archives" on page 25-7.

25.3.7 Creating a User Interface Master Template

You can create a UI Master Template from scratch or copy, rename, and then modify one of the predefined UI Master Templates.

The predefined UI Master Templates are located in the Master Templates Folder in the Main area of the Repository. For more information, see Section 20.2, "User Interface Master Templates" on page 20-2.

To create a UI Master Template based on one of the predefined Templates, create a copy of the predefined template, then modify it as necessary. Copying entities is described in Section 25.5.1 on page 25-9.

To create a UI Master Template from scratch:

1. From the Main area of the Repository, in the same row as the Folder in which you want to store the template, click the icon in the **Create** column.
2. Select an Object Type of **UI Master Template**, and then click **Continue**.
3. Optionally modify the default **Name**, enter a **Description**, and any **Notes** about the template.

   The template name must be unique within the Folder in which you are creating it.
4. Optionally modify any default settings, such as the template’s navigation style, Pagination and Layout settings, or which UI content Template are used to display Model structure.

   For details, see Section 20.2.2, "UI Master Template Information and Settings" on page 20-4.
5. Click **Apply**, or **Apply and Create Another**.

25.4 Opening Objects For Editing

This section describes how to modify an objects’s definition or value. To learn how to rename, move, copy, and delete objects, see Section 25.5, "The Main Area of the Repository Actions List" on page 25-9.

To open an object for editing:

1. In the Main area of the Repository, locate the object.
2. Click the icon in the **Edit** column.

   For details about modifying specific types of objects, see:

   - Modifying Models
   - Modifying Properties
   - Modifying Effectivity Sets and Usages

- Modifying Configurator Extension Archives
- Modifying the Archive Path for a Model
- Editing a User Interface Content Template (Chapter 31)
- Editing a User Interface Master Template (Chapter 31)

25.4.1 Modifying Models

When you open a Model for editing from the Main area of the Repository, general information about the Model appears in the General area of the Workbench. For details, see Chapter 28, "General Area of the Workbench".

By default, you must lock a Model before modifying it. For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.

To modify the Model's structure, rules, or UI:

1. Lock the Model by clicking the icon in the **Locking** column.

   For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.
2. Open the Model for editing by clicking the icon in the **Edit** column.
3. Navigate to the corresponding workbench by clicking one of the links at the top of the page.

For more information, see:

- Chapter 29, "Structure Area of the Workbench"
- Chapter 30, "Rules Area of the Workbench"
- Chapter 31, "User Interface Area of the Workbench"

25.4.2 Modifying Properties

For general information about Properties, see Chapter 5.

This section describes how to modify a Property and apply the change to all nodes assigned to that Property. You can do this only for Properties that were created in Configurator Developer.

If you want to modify the value of a Property assigned to a specific node, open the node for editing in the Structure area of the Workbench. For details, see Section 29.16, "Modifying Structure Node Properties" on page 29-9.

To modify a Property that may be shared by multiple nodes:

1. In the Main area of the Repository, in the same row as the Property you want to edit, click the icon in the **Edit** column.

   If a Property was imported from Oracle Bills of Material, the icon in the **Edit** column is read-only.
2. Modify the Property's **Name**, **Description**, **Data Type**, or **Default Value**, then click **Apply**.

25.4.3 Modifying Effectivity Sets and Usages

You can modify Effectivity Sets and Usages in the Main area of the Repository.

For general information about Effectivity Sets and Usages, see Chapter 6, "Effectivity".

25.4.3.1 Modifying Usages

You cannot modify the name of a Usage if it is assigned to an existing publication. You can modify the description of a Usage at any time.

25.4.3.2 Modifying an Effectivity Set

You can modify the Name and Description of an Effectivity Set at any time.

Before modifying an Effectivity Set's date range, use the List Effectivity Set Members actions to see which objects use it. For details, see Section 25.5.5, "List Effectivity Set Members" on page 25-12.

You cannot modify an Effectivity Set's date range if it is used by rule in a Rule Sequence and Configurator Developer is unable adjust the dates of the other rules in the sequence to accommodate the new dates. In this case, Configurator Developer displays a message that includes the name of the Rule Sequence.

25.4.4 Modifying Configurator Extension Archives

The Main area of the Repository provides pages for modifying and viewing the contents of Configurator Extension Archives. See Section 25.3.6, "Creating a Configurator Extension Archive" on page 25-4 for information on creating a Configurator Extension Archive.

On the Configurator Extension Archive page, you can define the contents of your Archive using either of the following methods:

- Referencing a Java class archive file through a URL
- Uploading a Java class archive file to the database

See Section 17.3, "Configurator Extension Archives" on page 17-3 for information on the characteristics of each method.

1. This procedure assumes that you have just created a Configurator Extension Archive, as described in Section 25.3.6 on page 25-4.
2. In the Main area of the Repository, locate the Configurator Extension Archive that you want to modify, then click the icon in the **Edit** column in the same row as the Archive.
3. On the Configurator Extension Archive page, click **Specify Java Class Archive**.
4. On the Specify Java Class Archive page, use one of the following methods for the **New Archive**:
   - To reference the Java class archive file through a URL, select **URL** and enter the full URL in the text field, as in the following example:

     ```
     http://server.com/myarchive.jar
     ```

     The URL is not verified by Configurator Developer, so be sure that you have entered it accurately.
   - To upload the Java class archive file to the database, select **Upload**, click **Browse**, then use the resulting file chooser dialog to navigate to and select your Java archive file. The file must be on a locally mounted file system.

     If you subsequently change this specify this Archive to specify the Java class archive through a URL, then the Archive's previously uploaded data is deleted from the database.
5. Click **Apply**.

After you have specified the contents of a Configurator Extension Archive, using either of the preceding methods, the Configurator Extension Archive page displays a specification of the **Archive File**, with one of the following meanings:

- For a Java class archive file that was uploaded to the Oracle Configurator schema from a locally mounted file system, the name of the original archive file is displayed (such as `myarchive.jar`). Because the archive now resides in the database, the file system path to the archive file is no longer relevant.
- For a Java class archive file that is referenced through its URL, the full URL is displayed. If the Configurator Extension Archive is subsequently placed in the Archive Path of a Model, then Oracle Configurator accesses the archive through this URL. However, before such a Model is published, the Archive must be uploaded or the connection between the Archive and the Model will be lost, and the publication process will fail when you run one of the publication concurrent programs.

Complete the definition of the Configurator Extension Archive by performing the following steps.

6. Click **Apply** to apply the definition of your Configurator Extension Archive.

   You are returned to the Main area of the Repository.

7. You can now add this Configurator Extension Archive to the Archive Path for a Model. See Section 28.10.2, "Adding Archives to a Model's Archive Path" on page 28-7 for details.

To check the contents of your Configurator Extension Archive, see Section 25.4.4.1, "Viewing the Classes in an Archive" on page 25-8.

25.4.4.1 Viewing the Classes in an Archive

You can view the contents of a Configurator Extension Archive, to determine which Java classes it contains.

1. In the Main area of the Repository, locate the Configurator Extension Archive that you want to modify.
   - To view the classes in the Archive without editing the definition of the Archive, click the name of the Configurator Extension Archive, which is a link to a read-only version of the Configurator Extension Archive page. On this read-only page, you can then click **Edit** to modify the Archive, as described in Section 25.4.4, "Modifying Configurator Extension Archives" on page 25-7.
   - To edit the definition of the Configurator Extension Archive as well as view its classes, click the icon in the **Edit** column in the same row as the object.
2. On the Configurator Extension Archive page, click **View Contents**.

   The resulting View Contents page displays the Name and Description of the Configurator Extension Archive, and the specification of the source Java archive file.

   A table displays the Java classes contained in the Configurator Extension Archive.
3. Expand the table to display the names of the Java classes.
   - Each Java class in the Archive is represented by a row in the table.
   - The hierarchy of packages in the Java archive, if any, is represented by the hierarchy of the table. Each package is represented by a row in the table that includes a **Focus** control and a hide/show toggle. Click the hide/show toggle to view the next level of the package. To view the complete hierarchy of the

archive, click **Expand All**. To narrow the focus of the view to a particular package, click its **Focus** control. To expand the focus again, click on a package name in the locator links in the table heading.

- If the Java archive file contains any Java source code files in addition to compiled class files, the names of both kinds of files are displayed. However, only class files can be selected.

Note: Do not confuse the list of classes in a Configurator Extension Archive, described here, with the list of bindable classes for a Model. The latter is described in Section 30.11.2, "Choosing the Java Class" on page 30-11.

---

4. When you are finished viewing the contents of the Configurator Extension Archive, click **Configurator Extension Archive** in the locator links.

25.5 The Main Area of the Repository Actions List

The actions available in the **Actions** list of the Main area of the Repository perform common operations on all selected objects in the hierarchical table.

These actions include:

- Moving and Copying Objects
- Deleting Objects
- Renaming Objects and Modifying Descriptions
- List Referencing Models
- List Effectivity Set Members

25.5.1 Moving and Copying Objects

Use the Move action to move an object from one location in the Main area of the Repository to another. Use the Copy action to create a copy of an object. When you copy or move an object, you must also specify a destination for the new object. By default, Configurator Developer creates the copy of the object(s) in the same location as the original object, but you can specify a different Folder as the destination.

When you copy an object, Configurator Developer gives the new copy a unique name. For example, if you copy an Effectivity Set called ”MyEffectivitySet," the copy is called "Copy (1) of MyEffectivitySet." If this name exists, then Configurator Developer increments the number in the new name. For example: Copy(2) of MyEffectivity Set.

Copying a Folder copies everything that the Folder contains (for example, Models, subFolders, Usages, and so on).

When copying a Model or a Folder that contains Models, you must also specify how you want to copy Model References. Select one of the following:

- **Maintain Existing References**: Select this option if you do not want to create new copies of all Referenced Models included in your selection.
- **Copy Entire Reference Chain**: Select this option if you want to create new copies of all Referenced Models included in your selection.

To copy or move one or more objects in the Main area of the Repository:

1. Select the object(s) you want to copy or move.

2. Select **Copy** or **Move** from the **Actions** list, and then click **Go**.
3. Specify how you want to copy Model References (if applicable), and select a destination for the object.

   If you chose **New Folder** as the destination, click **Select Location**, select a Folder from the list, then click **Apply**.
4. Click **Apply**.

> **Note:** By default, you can copy a Model or UI Content Template only if you have it locked, or it is not locked. For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.

25.5.2 Deleting Objects

You can delete an object only if it is not being used or referred to by another object. For example, you cannot delete a Model if it another Model references it, and you cannot delete a Usage or Effectivity Set if they are currently in use.

Deleting an object also deletes all of its child objects. For example, deleting a Folder deletes all of the objects that it contains.

> **Note:** By default, you can delete a Model or UI Content Template only if have it locked or it is not locked. For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.

To delete any objects that appear in the Main area of the Repository:

1. Select the object(s).
2. Select **Delete** from the **Actions** list, and then click **Go**.
3. Click **Yes** to confirm the action.

Refer to the following sections for more information:

- Section 25.5.2.1, "Deleting Models" on page 25-10
- Section 25.5.2.2, "Deleting Effectivity Sets and Usages" on page 25-11
- Section 25.5.2.3, "Deleting Properties" on page 25-11

25.5.2.1 Deleting Models

You can delete a Model only if all of the following are true:

- The Model is not referenced by another Model. For details about References, see Chapter 4.

  To see if a Model is referenced by other Models, select it in the Main area of the Repository, then select List Referencing Models from the Actions list. See Section 25.5.4, "List Referencing Models" on page 25-11.
- No active publications exist for the Model.

  For more information about Publishing, see Section 23.1 on page 23-1.
- The Model is not locked by another user.

  See Section 24.2, "Locking Models and UI Content Templates" on page 24-5.

25.5.2.2 Deleting Effectivity Sets and Usages

You can delete an Effectivity Set only if it is not assigned to any Model structure nodes or rules. To see all components to which an Effectivity Set is assigned, select it in the Main area of the Repository, then select List Effectivity Set Members.

To delete an Effectivity Set that is in use, you must first remove the association between it and the object. To do this, open the associated object for editing in the appropriate Workbench, then modify the Effectivity setting (for example, select Set Always Effective) and then click Go.

25.5.2.3 Deleting Properties

You can delete only Properties that you create in Configurator Developer. Properties imported with a BOM Model, BOM Option Class, or BOM Standard Item are read-only and cannot be modified or deleted.

To remove a Property's association with a Model node, see Section 29.15, "Removing Properties from a Model Node" on page 29-9.

25.5.3 Renaming Objects and Modifying Descriptions

You can rename an object or update its description at any time.

Configurator Developer does not display an error or warning message when you edit an object's name or description because this change does not adversely affect the availability of Model components. The change propagates to all elements that share that object.

To rename an object:

1. Select one or more objects.
2. Select **Rename** from the **Actions** list, and then click **Go**.
3. Enter the new name(s), and then click **Apply**.

Alternative method:

1. In the same row as the object you want to rename, click the icon in the **Edit** column.
2. If you selected a Model, the General area of the Workbench appears. In this case, click **Edit Details**.
3. Enter the new name, and then click **Apply**.

To update an object's description:

1. In the same row as the object you want to rename, click the icon in the **Edit** column.
2. If you selected a Model, the General area of the Workbench appears. In this case, click **Edit Details**.
3. Modify the object's description, and then click **Apply**.

25.5.4 List Referencing Models

Use the List Referencing Models action to see which Models reference the Model(s) you select.

For general information about References, see Chapter 4.

To list all referencing Models:

1. In the Main area of the Repository, select one or more Models.
2. From the **Actions** list, select **List Referencing Models**.

   The View Referencing Models page lists each Model that references the Model(s) you selected.
3. Click **Return to Repository Objects**.

25.5.5 List Effectivity Set Members

Use the List Effectivity Set Members action to see all nodes or rules that use a specific Effectivity Set, or multiple Effectivity Sets.

For general information about Effectivity Sets, see Section 6.3 on page 6-2.

To see all nodes or rules that use an Effectivity Set:

1. In the Main area of the Repository, select one or more Effectivity Sets.
2. From the **Actions** list, select **List Effectivity Set Members**, and then click **Go**.

   The Effectivity Set Members page lists each node and rule that uses the Effectivity Set(s) you selected. The name of each object appears as a link.
3. Optionally click the object's name to view its details, and then click Edit if you want to modify the object.
4. Click **Return to Repository Objects** to return to the Main area of the Repository.

26 Item Master Area of the Repository

This chapter describes the various tasks you can perform in the Item Master area of the Repository.

This chapter includes the following sections:

26.1 Introduction

Use the Item Master area of the Repository to manage imported and manually created Items and Item Types that are stored in the CZ schema's Item Master.

For a general description of the CZ schema's Item Master, Items, and Item Types, see Chapter 2. For details about Properties, see Chapter 5.

26.2 Creating a New Item Type

To create a new Item Type:

1. Navigate to the Item Master area of the Repository.
2. Click **Create Item Types**.
3. Enter a **Name** and **Description** of the new Item Type.
4. To add Properties to the new Item Type:
   - Click **Manage Properties**.
   - Select the Properties to add, then click **Add to Selected List**.
   - Enter a **Default Value**, then click **Apply**.
5. Click **Apply**.

26.3 Creating a New Item

To add a new Item to an Item Type:

1. Navigate to the Item Master area of the Repository.
2. In the same row as an existing Item Type, click the icon in the **Create** column.
3. Enter a **Name** and **Description**.
4. Select **Orderable** if you want the Item to appear in the Summary page when it is added to the configuration at runtime.

   For more information about this setting, see Section 2.3 on page 2-2.
5. Enter any additional information about the Item in the **Notes** field.
6. Click **Apply**, or **Apply and Create Another**.

26.4 Changing the Item Type of an Item

You can change the Item Type to which an Item belongs only if the Item was created in Configurator Developer.

To change an Item's Type:

1. Navigate to the Item Master area of the Repository.
2. Expand an Item Type to view its Items, focus in on the Item Type, or click **Expand All** to view all Items.
3. In the same row as the Item you want to modify, click the icon in the **Edit** column.
4. In the Item's details page, click **Choose**.

   If this button is read-only, the Item you are modifying is imported and you cannot select a different Item Type.
5. Select an Item Type, then click **Apply**.
6. In the Item's details page, click **Apply**.

   The Item appears as a child of the Item Type you selected in step 5.

26.5 Editing Properties Assigned to Items or Item Types

When editing an Item, you can change the value of any Properties that were assigned to the Item in Configurator Developer, or remove a Property's association with the Item. This is true even if the Item or the Property was imported from Oracle Bills of Material. However, if the Property was imported with the Item, the Property's value is read-only and you cannot modify it when editing the Item or remove the Property when editing the Item Type.

When you create nodes in the Model structure either by adding Items or Item Types from the Item Master or by running a Populator, Configurator Developer maintains a relationship between the Properties on the nodes in the Model structure and the corresponding Items in the Item Master. Therefore, if you modify a Property's value in the Item Master, Configurator Developer updates the corresponding value in the Model structure (although the reverse is not true). However, if you change the Property's value in the Model structure, any future changes that you make to the Property in the Item Master will not be reflected in the Model structure.

To modify the value of a Property assigned to an Item:

1. Navigate to the Item Master area of the Repository.
2. In the same row as the Item you want to modify, click the icon in the **Edit** column.
3. Modify the Property **Value**, or select it, and then click **Reset to Default Value**.

   The default value is the value the Property had when it was added to the Item in Configurator Developer.
4. Click **Apply**.

26.6 Adding Properties to Items and Item Types

For background information about Properties, see Chapter 5, "Properties".

You can assign any User Property to an Item Type, regardless of whether the Item Type or Property was imported or was created in Configurator Developer. When you assign a Property to an Item Type, Configurator Developer assigns the Property to all of the Item Type's child Items. You can then modify the Property's value for specific Items, if required. This is explained in Section 26.5 on page 26-2.

When editing an Item, you can modify any associated Property values, regardless of whether the Properties were imported or were created in Configurator Developer.

To add a Property to an Item Type:

1. In the Item Master area of the Repository, in the same row as an Item Type, click the icon in the **Edit** column.
2. In the Item Type's details page, click **Manage Properties**.
3. Select one or more Properties, then click **Add to Selected List**.
4. Click **Apply**.
5. In the Item Type's details page, click **Apply**.

   Configurator Developer assigns the Property to all of the Item Type's child Items.

26.7 Deleting an Item or Item Type

You can delete only Items and Item Types that were created in Configurator Developer. In other words, you cannot delete imported Items or Item Types.

Additionally, you can delete an Item Type only if it does not contain any Items. If you attempt to delete an Item Type that has Items, Configurator Developer prompts you to delete its child Items first.

To delete an Item or Item Type:

1. In the Item Master area of the Repository, select the Item or Item Type.
2. Select **Delete** from the **Actions** list, and then click **Go**.
3. Click **OK**.

26.8 The Item Master Area of the Repository Actions List

The Actions list in the Item Master Repository includes Delete and Rename. The procedures for deleting or renaming Item Master objects are the same as deleting or renaming objects in the Main area of the Repository.

For details, see:

- Section 25.5.2, "Deleting Objects" on page 25-10
- Section 25.5.3, "Renaming Objects and Modifying Descriptions" on page 25-10

27 Publications Area of the Repository

This chapter describes the tasks you can perform in the Publications area of the Repository.

This chapter includes the following sections:

27.1 Introduction

For general information about publishing, see Chapter 23, "Publishing".

In the Publications area of the Repository, all Model publications appear alphabetically by Model name. Depending on the View you selected, the following information may appear:

Model/Publication ID: The name of the Model as it appears in the Main area of the Repository, and the Model publication ID number. The publication ID number is generated by the database when the publication is created.

In each row, the Model name is a parent of the publication ID. Therefore, you may need to expand the Model name row to view the publication ID, and then edit or create a copy of the publication.

User Interface: The User Interface associated with the publication. You can create multiple UIs for a Model in Oracle Configurator Developer, but a Model publication can have only one UI.

Database Instance: The database on which the publication exists.

Published: The date the publication was created.

Status: The current status of the selected publication. Values include Complete, Pending, Update Pending, Processing, and Error. See the *Oracle Configurator Implementation Guide* for more information.

Disabled: Indicates whether the publication is currently enabled. Disabled publications are not available to host applications.

27.1.1 Publishing Actions

This section describes the UI controls and actions you can perform in the Publications area of the Repository.

Create Model Publication: Click this button to create a new Model publication.

Creating a new Model publication is described in Section 27.2 on page 27-2.

Enable/Disable: Use these buttons to enable or disable an existing publication.

For details, see Section 27.8 on page 27-7.

Republish: Select an existing publication and then click this button to republish a Model.

Republishing is described in Section 23.4 on page 23-3.

Delete: Select an existing publication and then click this button to delete a publication from the database. This action does not remove any Model data from the CZ schema.

Deleting a publication is described in Section 27.7 on page 27-7.

Edit: Click the icon in the **Edit** column to modify the applicability parameters for an existing publication. This function changes the availability of the publication, but does not actually copy Model data to the target database.

Editing publications is described in Section 27.6 on page 27-6.

New Copy: Click the icon in the **New Copy** column to create a new Model publication that is based on an existing publication.

You may want to do this if, for example, you want to create a new publication that has applicability parameters that are similar to an existing publication.

Copying publications is described in Section 27.3 on page 27-4.

Note: Click your browser's **Refresh** button to update the Status column for the list of publications. The status of one or more publications may change when one of the publishing concurrent programs completes successfully. For details about the publishing concurrent programs, see the *Oracle Configurator Implementation Guide*.

27.2 Creating a New Model Publication

Use the procedure described in this section to create a Model publication, which is the first step in the process of making a configuration model available to one or more host applications. If you want to publish the same configuration model but specify different applicability parameters, you can create a copy of an existing publication and then modify its applicability parameters. This process is described in Section 27.3 on page 27-4.

Before you can create a new publication, the UI and logic for the configuration model must be up to date in Configurator Developer. To refresh a UI or regenerate logic, see Section 11.5.1 on page 11-6.

For an overview of the publishing process, see Section 23.1, "Introduction" on page 23-1.

Note: Although it is possible to create and maintain publications on separate databases (for example, a development and a production instance), configuration models that will be available to customers should be published only from a single development environment. For more information, see the *Oracle Configurator Implementation Guide*.

To create a new Model publication:

1. In Configurator Developer, generate logic and refresh the UI for the Model you want to publish.

   For details, see Chapter 28, "General Area of the Workbench".

2. Navigate to the Publications area of the Repository, and then click **Create Model Publication**.

3. Select the Model to publish, then click **Continue**.

4. In the publication details page, select a **User Interface** from the list.

   Leave this setting blank if you do not want to associate a UI with this publication. For more information, see Section 27.2.1, "Publishing a Model without a User Interface" on page 27-4.

   Note: For information about the **Product ID**, see the *Oracle Configurator Implementation Guide*.

   ---

5. Select a **Target Database Instance** from the list. This is the database from which you want the publication to be available to host applications. The list includes all Oracle Applications database instances defined in your organization, not just the instances that are currently enabled.

   Database maintenance tasks are described in the *Oracle Configurator Implementation Guide*.

   Note: If you are using Multiple Language Support (MLS), all installed languages on the source and target databases must be the same when publishing to a remote server. For more information, see Section B.8, "Publishing and Multiple Language Support" on page B-4.

   ---

6. Choose a publication **Mode** of either **Test** or **Production**.

7. Enter **Applicability Parameters**, including one or more **Applications**, **Languages, Usages**, and effective date range.

   For details, see Section 23.5, "Applicability Parameters" on page 23-4.

   Note: A publication's applicability parameters must be unique before you can create it. For details, see Section 23.6 on page 23-5.

   ---

8. Optionally specify a Standalone Container Page and a JSP Container.

- The **Standalone Container Page** setting controls the document that defines the layout of each page in the UI. The default setting is **Default Page Layout**. In this case, Configurator Developer uses czBlafPageLayout.jsp when you generate a User Interface.

  To specify another container page, select **Custom Page Layout** and enter the name of a document whose type is Page Layout.

- The **JSP Container** settings control which Java Server Page Configurator Developer uses as a container for each UI page. In other words, the page you specify wraps the content of each UI page at runtime.

  Select **Empty Container** to display UI content without the Oracle Configurator image that czBlafPageLayout.jsp provides.

  Accept the default value (**Container with Header Bar**) to display UI content without the Oracle Configurator image that czBlafPageLayout.jsp provides.

  To use a different JSP, select **Custom Container** and enter the name of the file. The file you specify must exist in the publication and user interface definition tables.

  For more information about how the runtime Oracle Configurator renders the UI, see the *Oracle Configurator Implementation Guide*.

9. Click **Apply**.

   Developer displays an error message if one or more applicability parameters overlaps with an existing publication. If this occurs, modify the parameters as required, then click **Finish**. The new publication appears in the list of publications with a unique publication **ID** number and a **Status** of Pending.

   Note: Make a note of the publication ID. This value is a required parameter when running one of the publishing concurrent programs that copies publication data to the target database.

10. Copy the publication to the target database.

    See Section 27.5, "Copying Model Data to a Database" on page 27-5.

27.2.1 Publishing a Model without a User Interface

Publish a configuration model without specifying a UI only if you plan to use another application to create a custom user interface that accesses the Configuration Interface Object (CIO). This implementation is not described directly in any Oracle Configurator documentation. For details about the CIO, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

If a host application launches Oracle Configurator to configure an item that has been published, but no user interface is available for the publication, the runtime Oracle Configurator displays an error similar to the following:

```
No user interface found for publication ID 456123.
```

27.3 Copying an Existing Model Publication

Instead of creating a new publication from scratch, you can copy an existing publication and then modify its applicability parameters. You may want to do this if, for example, you want to create an additional publication for a Model and make only minor changes to its applicability parameters.

To copy an existing publication:

1. Navigate to the Publications area of the Repository, then select an existing publication from the list.
2. Click the icon in the **New Copy** column.
3. Modify the new publication’s applicability parameters as required.

   Applicability parameters are described in Section 23.5 on page 23-4.
4. Click **OK**.

   Make a note of the publication ID for future reference.
5. Modify the settings in the **Container Page/Region** section, if required.

   For details, see Section 27.2 on page 27-2.
6. Click **Apply**.
7. Copy the publication to the target database.

   For details, see Section 27.5 on page 27-5.

27.4 Republishing a Model

For general information about republishing, see Section 23.4 on page 23-3.

To republish a Model:

1. Navigate to the Publications area of the Repository.
2. Select the publication to update, and then click **Republish**.

   Configurator Developer creates a concurrent request to copy Model data to the specified database. Make a note of the new publication ID for future reference.
3. Update the publication by copying the publication to the target database.

   For details, see Section 27.5 on page 27-5.

27.5 Copying Model Data to a Database

When you create a publication, Configurator Developer creates a new record in your development database (that is, the database on which Configurator Developer is running). This record consists of the entire configuration model, including Model structure, rules, and (typically) a UI. Before a host application can access the publication, this data must be copied to the database you specified as the "target" when creating the publication. The development and target database can be the same, but different tables are used to store the publication that you create in Configurator Developer and the one that is accessed by host applications.

You copy Model data to the target database (or different tables in the same database) by running one of the publishing concurrent programs. There are two publication concurrent programs: one selects only a single publication that you specify, while the other selects all publications that have a status of Pending. For more information about these programs, see the *Oracle Configurator Implementation Guide*.

The concurrent program selects the publication record(s) that you created in Configurator Developer and copies the new or changed data to the target database. When the program completes successfully, it creates a new publication record (if you are creating a new publication) or updates the existing publication (if you are republishing). At this point, the status of the publication changes to Complete in the Publications area of the Repository, and host applications can view the publication in a runtime Oracle Configurator.

All affected Models must either be unlocked or locked by you before running the concurrent program. For details, see Section 24.2.2, "Automatic Model Locking" on page 24-6.

Note: If you make any changes to the Model or any of its referenced Models in Configurator Developer before the concurrent program copies the data to the target database, the Model's logic will not be up to date and the program will fail. When this occurs, Configurator Developer sets the publication status to Error. To create the publication and change its status to Complete, you must republish the Model. This is described in Section 27.4 on page 27-5.

27.6 Editing a Model Publication

Edit a Model publication's applicability parameters when you want to change its availability to host applications. For example, you can add host applications to the Applications parameter to make it more widely available, modify the effective dates, or change the publication Mode. To make a publication available for a different application or Usage *and* for a different date range, you must create a new publication. For details, see Section 27.2 on page 27-2.

You can edit a publication only if its status is Complete or Pending. For more information about publication statuses, see the *Oracle Configurator Implementation Guide*.

Note: If a Model that contains UI Template References is published and you modify the template in Configurator Developer, the changes do not appear in any UIs that reference the template until you republish the Model. Republishing is explained in Section 27.4 on page 27-5. For more information, see Section 21.16.1, "User Interface Template References" on page 21-48.

Changing a publication's applicability parameters only affects its availability to host applications, it does not create a new publication in the database, copy Model data, or modify existing Models in any way.

Configurator Developer does not allow you to make changes that would create publications with overlapping applicability parameters in the same database instance. Overlapping applicability parameters are explained in Section 23.6 on page 23-5.

To modify a publication's applicability parameters:

1. Navigate to the Publications area of the Repository.
2. In the same row as the publication you want to modify, click the icon in the **Edit** column.
3. Modify the publication's applicability parameters. You can modify the Mode, Applications, Languages, Usages, and effective dates.
4. Modify the settings in the **Container Page/Region** section, if required.

For details about these settings, see Section 27.2 on page 27-2.

5. Click **Apply**.

Note: When you update a publication's applicability parameters in Configurator Developer, the changes propagate automatically to the publication in the target database. Therefore, running one of the publication concurrent programs is not necessary.

27.7 Deleting a Publication

Deleting a publication prevents any runtime Configurators from accessing the Model for any further testing or production activities. You may also want to delete a publication if, for example, one of the publication concurrent programs has an error status upon completion, and you want to republish it.

You can delete a publication only if its status is Complete, Pending, or Error.

Deleting a publication removes the publication record from the database, but it does not delete any data in the CZ schema that is associated with the publication. For information about purging publication records and data from a database, see the *Oracle Configurator Implementation Guide*.

To delete a publication:

1. Navigate to the Publications area of the Repository.
2. Select a publication, then click **Delete**.
3. Click **OK**.

Note: It is not necessary to run one of the publication concurrent programs to delete the publication record in the target database. This is because deleting a publication in Configurator Developer also deletes the publication that is accessed at runtime by host applications.

27.8 Enabling or Disabling a Publication

You might want to disable a publication if, for example, you need to make it temporarily unavailable to host applications.

To enable or disable a publication:

1. Navigate to the Publications area of the Repository.
2. Select a publication, then click **Enable** or **Disable**.
3. Click **OK** to confirm the action.

28 General Area of the Workbench

This chapter describes the tasks you can perform in the General area of the Workbench.

This chapter includes the following sections:

28.1 Introduction

The General area of the Workbench appears when you open a Model for editing or viewing from the Main area of the Repository. This area of the Workbench displays general information about the selected Model and allows you to perform Model-specific operations such as generating logic, refreshing User Interfaces, editing Configurator Extension Archive Paths, creating a Model Report, and running Populators.

To modify the Model's structure, rules, or UIs, navigate to the appropriate area of the Workbench by clicking the corresponding link at the top of the page.

28.2 Model Report

Generate a Model Report to create a portable document format (PDF) file containing detailed information about a configuration model. When submitting the report, you specify whether it includes details about the Model structure, User Properties, rules, Item Master data, and any referenced Models. You can generate and print a Model Report at any time during the configuration model development process.

You must have Oracle XML Publisher installed at your site to generate a Model Report. This is because Configurator Developer uses Oracle XML Publisher to generate and format the report's content.

To generate a Model Report:

1. Open a Model for editing.
2. In the General area of the Workbench, click **Model Report**.
3. In the Report Contents page, specify the data you want to include in the report. Choose from the following:
   - **Model Structure**: Select **Include** to include details about the Model structure in the report. Otherwise, select **Exclude**. For details, see Section 28.2.1, "Model Report Content and Column Headings" on page 28-2.

     Deselect **Include Property Data** to exclude the names and values of any User Properties from the report.
   - **Rules**: Select **Include All** to include all rules in the Model, or select **Include if Enabled** to include only rules that are currently enabled.

     Select **Exclude** if you do not want the report to provide details about any of the Model's rules.
   - **Item Master**: Choose whether to include Item Master data for the Model, or for the entire Repository (that is, the CZ schema).

     Select **Exclude** if you do not want the report to include any Item Master data.

     Note: The report may take a long time to complete if you include Item Master data for the entire Repository. You may want to generate a separate report that includes only Item Master data.

     Deselect **Include Property Data** if you do not want to list the names and values of any associated User Properties.
   - **Referenced Model Data**: Indicate whether the report criteria also applies to all of the Model's referenced (child) Models.

     For example, if you chose to include details about Model structure and rules, the report also contains details about the Model structure and rules for each referenced Model.
4. When you are satisfied with the report criteria, click **Generate Report**.
5. When the report is complete, a link appears at the bottom of the page. Click this link, and then choose whether to save the report locally (as a PDF file) or open it using Adobe Acrobat.

28.2.1 Model Report Content and Column Headings

A Model Report displays information in several sections that correspond to the criteria you specified when submitting the report. Within each section, the configuration model data appears in separate tables for the Model structure, Properties, rules, and Item Master data.

An empty table cell indicates that either the value does not apply to a specific node, or no value exists.

Model Structure Section

This section appears if you chose to include Model Structure when defining the report's content.

This section contains two sub-sections: Model Hierarchy View and Model Detail Views. The Model Hierarchy View lists only the root parent node and its children, which includes Components and referenced Models. The table in the Model Detail Views section also lists each Component and referenced Model, but shows all of their child nodes. This view is similar to viewing the Model structure in Configurator Developer when it is fully expanded. In other words, it contains details about all nodes in the Model, including BOM Option Classes, BOM Standard Items, Features, Options, and so on.

Following is a description of the various column headings that appear in the Model Structure section:

Level: The level at which the node appears in the Model structure. The root Model node is level 1, the root node's children are level 2, and so on.

Node Name: For details, see Section 29.17.1, "Name" on page 29-10.

Description: The node's description. See Section 29.17.2, "Description" on page 29-11.

Type: The node's type. For details, see Chapter 9, "Model Structure Node Types" on page 9-1.

Instances: The values in this column reflect the Instantiability settings defined for a Model Reference or Component node. For details, see Section 29.17.6, "Instances" on page 29-13.

Quantity: For BOM items, the Quantity column displays the node's minimum and maximum quantity. For example, "1/3" means a minimum quantity of 1 and a maximum quantity of 3. For details, see Section 11.3.1, "Quantity Cascade Calculations" on page 11-3.

For non-BOM items, this column refers to the Minimum and Maximum Selections values for an Option Feature, and the Minimum and Maximum values for Decimal and Integer Features.

Initial Value: Initial values are explained in Section 9.12 on page 9-6.

Properties: This column lists the name and value of all User Properties assigned to each node (if you chose to include Property data in the report).

For details, see Section 5.2, "User Properties" on page 5-1.

Item Master Section

This section appears only if you chose to include Item Master data when defining the report's content.

This section lists the name and description of each Item Type and Item used in the Model, or in the Repository (depending on the report criteria). It also displays the User Properties assigned to each Item Type and Item, if you chose to include Properties in the report.

For details about Item Types and Items, see Chapter 2, "The CZ Schema's Item Master" on page 2-1.

Rules Section

This section appears if you chose to include rules when defining the report's content.

This section contains two main sections: Folder Hierarchy View and Rule Folder Views. The Folder Hierarchy View section displays the names and descriptions of all of the Model's Rule Folders and, if you chose to include referenced Model data, any referenced Model Rule Folders.

A "1" in the Level column indicates that the Folder is a top level Folder for either the parent Model or a referenced Model. Levels 2, 3, and so on indicate sub-Folders.

The Rule Folder Views section provides details about each Rule Folder listed in the Folder Hierarchy View, such as the rules that each Folder contains and each rule's name, type, and definition. All rules or only enabled rules may be included, depending on the criteria you specified when submitting the report.

The rule definitions for Logic, Numeric, Property-based Compatibility Rules appear in the Rule Text column, and are displayed in the Constraint Definition Language (CDL).

Because Explicit Compatibility Rules and Design Charts cannot be expressed in CDL, their definitions appear separately, in tabular format. Functional Companions and Configurator Extensions are also displayed in a separate table.

For details about CDL, see the *Oracle Configurator Constraint Definition Language Guide*.

28.3 Model Details

This section lists the Configurator Developer user who created the selected Model, and the date the Model was originally created. It also indicates when any part of the Model was last modified, and the Configurator Developer user who made the changes.

If the selected Model is a BOM Model, or is a non-imported Model that references a BOM Model, the Product Key is read-only, and consists of the BOM Model's Oracle Inventory Item ID followed by its Inventory Organization ID. For example, 452:1534.

If the selected Model was created in Developer, or was imported from a source other than Oracle Bills of Material, and does not contain any BOM Models, you can enter or modify the Product Key by clicking Edit Details.

To modify the Model name, description, or related notes, click Edit Details. To open the Model for editing, navigate to the Structure area of the Workbench.

28.4 Base Inventory Item Details

If the selected Model is an imported BOM Model, this section lists information about the Oracle Inventory Item on which the Model is based. If a responsibility that provides access to Item Management is assigned to your Oracle Applications user name, you can navigate to Item Management to view additional details about the Item in that application.

For example, if Item Manager is one of the responsibilities that is assigned to your user name, you can click the button provided to navigate to Item Management.

For more information about Item Management, refer to Oracle Item Management documentation.

28.5 Bill of Materials Status

If the Model you are viewing was imported from Oracle Bills of Material, this section indicates the last time the BOM Model data was updated. You must refresh a BOM Model whenever the BOM Model changes in Oracle Bills of Material. For details about refreshing a BOM data, see the *Oracle Configurator Implementation Guide*.

For more information about BOM Models, see Section 3.3 on page 3-2.

28.6 Populator Status

This section indicates the last time any Populators defined for the Model were executed and whether the Model needs to be repopulated. A Model must be repopulated when Items or Item Types have been added to or changed in the CZ schema’s Item Master.

Click Repopulate to run all Populators defined for the selected Model.

For more information about Populators, see Chapter 10.

28.7 Logic Generation Status

This section indicates the last time logic was generated for the selected Model and whether logic needs to be regenerated. For example, you must regenerate logic whenever you define new or modify existing rules. It is also recommended that you generate logic before unit testing the Model.For more information, see Section 11.5.1, "Generating Logic" on page 11-6.

To update logic for the selected Model, click Generate Logic. A status message indicates whether logic was generated successfully with Warnings, or if it failed with errors. Refer to the following sections for details.

28.7.1 Logic Generation Warnings

It is possible for logic to be generated successfully but still display warning messages. In this case, you can unit test the configuration model.

The logic generation status message includes warnings when:

- A rule is incomplete or defined incorrectly (in this case, the rule is ignored when unit testing)
- Model structure is missing or a node is defined incorrectly

Examples of logic generation warning messages:

- "Compatibility rule must have at least two participating features, rule ‘Test Exp Rule“ ignored"
- "The Model structure has changed and the rule ‘My Test Rule’ now contains the deleted node ‘Color Feature’. "

28.7.2 Logic Generation Errors

The logic generation status message includes errors when Model structure is missing or invalid. You cannot unit test or publish a configuration model until all of the errors caught by logic generation are resolved.

Examples of logic generation errors include:

- “Node ‘SI1142’ must allow decimal quantity since its parent ‘OC5231’ allows decimal quantity.”
- “Invalid Model structure: Multiple references exist to the trackable instance ‘CN4321Z’ ”
- "Project does not exist for the specified ID: ‘512342’. No logic generated.’ ”

28.8 UI Refresh Status

This section lists any User Interfaces that need to be refreshed. A UI must be refreshed when the Model structure changes. For details, see Section 19.2.1, "Changes that Require a User Interface to be Refreshed" on page 19-3.

To refresh all UIs that are out of date, click Refresh UIs.

You can also refresh one UI at a time in the User Interface area of the Workbench. For details, see Section 31.3.11, "Refreshing a User Interface" on page 31-28.

Note: Configurator Developer does not update a UI if the Refresh Enabled setting is set to No in the UI's UI Definition. This is true even if the Model structure has changed. See Section 19.2.4, "The Refresh Enabled Setting" on page 19-8.

28.9 Runtime Display Names

This section indicates how the runtime Oracle Configurator generates the default UI captions for BOM and non-BOM Model structure nodes. To modify how Configurator Developer generates these captions, click Edit Display Names. You can create default UI captions using node names, descriptions, a User Property, or a text expression that you enter.

The default Display Name setting for BOM nodes is Description, while the default for non-BOM nodes is Name. If you select Description, be sure that all nodes in the Model have a description; otherwise, some UI elements may appear without a caption at runtime.

The Display Name setting determines each node's `DisplayName` System Property. When you generate a UI, the Text Source setting in each UI element's details page is set to "Display Name" by default; this means the runtime UI will generate the caption using the associated Model node's `DisplayName` System Property. For example, if you set Display Name to Description for BOM nodes, and you do not modify the default Text Source for any UI elements after generating a UI, BOM node descriptions are used as UI captions for all elements that represent BOM items at runtime.

For more information about System Properties, see Section 5.3 on page 5-2. The Text Source setting is described in Section 21.12, "User Interface Element Captions and Details" on page 21-35.

You can also create default UI captions using a User Property or a text expression that you enter. The User Property setting is not recommended unless all nodes in the Model share a common Property. Additionally, specifying a User Property whose value can change at runtime is not recommended because instance names may not appear as expected when the Property's value changes and new instances are created. For details about User Properties, see Section 5.2, "User Properties" on page 5-1.

A text expression can include System and User Properties and any text you enter. For more information, see Section 21.12.1, "Defining a Text Expression" on page 21-36.

If you are implementing Multiple Language Support (MLS), set Display Name to Description for both BOM and non-BOM nodes. For more information about MLS, see Appendix B, "Multiple Language Support" on page B-1.

28.10 Configurator Extension Archive Path

Use this section to edit the Archive Path for the selected Model. A Model's Archive Path affects the execution of the Configurator Extensions that are defined in the Model. For information on Configurator Extension Archives and the Archive Path, see:

- Section 17.3, "Configurator Extension Archives" on page 17-3
- Section 25.3.6, "Creating a Configurator Extension Archive" on page 25-4
- Section 17.3.1, "The Archive Path" on page 17-3

28.10.1 Editing a Model's Archive Path

To add Archives to a Model's Archive Path, or modify the current Archive Path, you must edit the Archive Path.

1. In the Main area of the Repository, locate the Model that you want to associate with a Configurator Extension Archive.
2. In the same row as the Model, click the icon in the **Edit** column.

   In the General area of the Workbench, the Archive Path for the Model appears in the **Configurator Extension Archive Path** section. For an explanation of the Archive Path, see Section 17.3.1 on page 17-3.

   You can click a name in the **Archive Name** column to go to the main page for redefining the selected Archive.
3. Click **Edit Archive Path**.

   The Edit Archive Path page lists all Configurator Extension Archives in the Main area of the Repository, subject to the current focus level. For each Archive, the list includes the name and location of the corresponding Java class archive file. If the Archive's classes have been uploaded to the database, then the **Archive Location** column shows the name of the uploaded Java class archive file, and the **Uploaded** column contains a mark.

   - To add Archives to the Archive Path, see Section 28.10.2, "Adding Archives to a Model's Archive Path" on page 28-7.
   - To modify the Archive Path, Section 28.10.3, "Modifying the Archive Path for a Model" on page 28-8.

28.10.2 Adding Archives to a Model's Archive Path

Before you can bind a Java class to your Model, you must add the Configurator Extension Archive containing that class to the Archive Path of the Model.

1. On the Edit Archive Path page, select the Configurator Extension Archive that you want to add to the Archive Path, then click **Add to Selected List**.

   When you do this, the selected Configurator Extension Archives appear in the Selected List. The most recently added Archive appears at the end of the list, meaning that it occurs at the end of the Archive Path.
2. Continue to add Archives to the Selected List, as desired.

   If you need to change the list of Archives in the Archive Path, see Section 28.10.3, "Modifying the Archive Path for a Model" on page 28-8.
3. When you are satisfied with your definition of the Archive Path, click **Apply**.

When you add an Archive to the Archive Path of a Model, that fact is reflected on the Configurator Extension Archive page for that Archive, where there is a table listing **Models Referencing This Archive**. This table contains links to the Models that include the Archive on their Archive Path. You can click a link to go directly to the General Workbench page for the Model, where you can edit its Archive Path. This is convenient when you change the classes in an Archive, and wish to make changes to the order in which the classes are loaded by the Configurator Extension Rules of the Models that use those classes.

28.10.3 Modifying the Archive Path for a Model

The Edit Archive Path page provides the **Selected List**, which lists the Archives that currently constitute the Archive Path for the Model.

- The Selected List is numbered, to indicate the search order among the Configurator Extension Archives.
- The **Path** column in the Selected List displays the path to the Configurator Extension Archive in the Main area of the Repository.

Use the Selected List to remove, reorder, or add Archives to the Archive Path.

- To remove one or more of the Archives from the Selected List, select them, then click **Remove**.
- To change the order of the Archives in the Selected List, click **Reorder**.

  The Reorder Archives page appears. To move an Archive higher or lower in the Selected List, select it and click the arrow controls.

  The order of the Selected List affects the Archive Path. See Section 17.3.1, "The Archive Path" on page 17-3 for details on why you might need to change the order of the Archive Path.
- When you are satisfied with your modifications to the Archive Path, click **Apply**.

29 Structure Area of the Workbench

This chapter describes the tasks you can perform in the Structure area of the Workbench.

This chapter includes the following sections:

29.1 Introduction

The Structure area of the Workbench displays the hierarchical structure of the Model you opened for editing in a table. You use this area of the Workbench to create and modify Model structure. To modify the type of information displayed for Model structure nodes, modify the default View, or create a new one. For details, see Section 24.1.1, "Views" on page 24-2.

For information and tips on designing Models for maximum runtime performance, refer to the following documentation:

- *Oracle Configurator Modeling Guide*
- *Oracle Configurator Performance Guide*

Note: When creating new Model structure, enter node names that are unique, meaningful to the Oracle Configurator end user and, if possible, descriptive of their intended purpose.

29.2 Creating a Component

For general information about Components, see Section 9.6 on page 9-2.

To create a Component:

1. In the same row as a Model or Component node, click the icon in the **Create** column.
2. Select **Create basic nodes**, and then select **Component** from the **Node Type** list.

   You can also create Components by selecting Create Item-based Nodes. For details, see Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4.
3. Click **Continue**.
4. Enter a **Name** and optionally a **Description** of the Component.
5. Enter values or define settings for the following as required:
   – "Display in User Interface" on page 29-11
   – "Instances" on page 29-13
   – "Properties" on page 29-13
   – "Populators" on page 29-13
   – "Effectivity" on page 29-14
   – "Notes" on page 29-15
6. Click **Apply**, or **Apply and Create Another**.

29.3 Creating a Feature

For general information about Features, including a description of each type of Feature, see Section 9.7 on page 9-2.

To create a Feature:

1. In the same row as a Model or Component node, click the icon in the **Create** column.
2. Select **Basic Nodes**, and then select **Option Feature, Integer Feature**, **Decimal Feature**, **Boolean Feature**, or **Text Feature** from the **Node Type** list.

   You can also create Features by selecting Create Item-based Nodes. For details, see Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4.
3. Click **Continue**.
4. Enter a **Name** and optionally a **Description** of the Feature.
5. Enter values or modify the default settings in the **Definition** section as required.

   For details about each setting, see "Definition" on page 29-12.

   If you are creating a Text Feature and want to require the end user to enter text at runtime, select the **Required** check box.

At runtime, an image appears next to the Feature to indicate that it is required. For details, see Section 20.2.2.7.2, "Status Indicator Images" on page 20-10.

6. Enter values or define settings for the following settings as required:
   - "Display in User Interface" on page 29-11
   - "Properties" on page 29-13
   - "Populators" on page 29-13
   - "Effectivity" on page 29-14
   - "Notes" on page 29-15
7. Click **Apply**, or **Apply and Create Another**.

29.4 Creating an Option

For general information about Options, see Section 9.8 on page 9-5.

To create an Option:

1. In the same row as a Feature node, click the icon in the **Create** column.
2. Select **Basic Nodes**, and accept the default **Node Type** of **Option**.

   You can also create Options by selecting Create Item-based Nodes. For details, see Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4.
3. Click **Continue**.
4. Enter a **Name** and optionally a **Description** of the Option, then click **Finish**.
5. Enter values or define settings for the following as required:
   – "Display in User Interface" on page 29-11
   – "Properties" on page 29-13
   – "Effectivity" on page 29-14
   – "Notes" on page 29-15
6. Click **Apply**, or **Apply and Create Another**.

29.5 Creating a Total or Resource

For general information about Totals and Resources, see Section 9.9 on page 9-5.

To create a Total or Resource:

1. In the same row as a Model or Component node, click the icon in the **Create** column.
2. Select **Basic Nodes**, then select a **Node Type** of either **Total** or **Resource**.

   You can also create Totals and Resources by selecting Create Item-based Nodes. For details, see Section 29.6, "Building Model Structure Using Items and Item Types" on page 29-4.
3. Enter a **Name** and optionally a **Description** of the node.
4. Enter values or define settings for the following as required:
   – "Display in User Interface" on page 29-11
   – "Initial Values" on page 9-6

   - "Properties" on page 29-13
   - "Violation Message" on page 30-18 (Resources only)
   - "Effectivity" on page 29-14
   - "Notes" on page 29-15

5. Click **Apply**, or **Apply and Create Another**.

29.6 Building Model Structure Using Items and Item Types

Use the procedure described in this section to build Model structure using data in the CZ schema's Item Master. For background information, see Chapter 2, "The CZ Schema's Item Master". By default, Model structure nodes that you create using this procedure have the same name, description, Properties, and so on as the Items you use to create them.

You can also build Model structure using data in the CZ schema's Item Master by defining a Populator. Populators are described in Chapter 10.

Items that you create from Item Master data can be added to a configuration in a runtime Oracle Configurator. However, Oracle Configurator does not return these items to the host application for ordering. This is true even if the items were created using imported BOM items.

To create a structure from Items or Item Types in the Item Master:

1. Open a Model for editing in the Structure area of the Workbench.
2. In the same row as a Model, Component, or Feature node, click the icon in the **Create** column.
3. Select **Create item-based nodes**, and then select the type of node you want to create from the list.
4. Click **Continue**.
5. Select the Item(s) or Item Type(s) to use as the source of the nodes you are creating, and then click **Finish**.

   The new node appears as a child of the Model, Component, or Feature you selected in step 2.

29.7 Reordering Model Structure

When you create nodes in Configurator Developer, they appear in the order in which they were created. For example, when you create a Feature Option, it appears below the Feature's existing Options. Imported BOM nodes appear in the order in which the BOM Model was defined in Oracle Bills of Material. The structure of an imported BOM Model cannot be modified in Configurator Developer, with one exception: you can modify the order in which BOM Model References appear in the structure. In other words, if a BOM Model has more than one Reference to other BOM Models, you can change the order in which the References appear in the Model structure.

When viewing a Model's structure in the Structure area of the Workbench, an icon appears in the Reorder Children column for each non-BOM node that has children. This includes Models, Components, and Option Features.

To reorder a node's children:

1. Open a Model for editing in the Structure area of the Workbench.

2. In the same row as the parent node, click the icon in the **Reorder Children** column.

   If the selected node has no children, Configurator Developer displays an error.

3. Select a node in the Children Order list, and then use the arrow icons to change its position.

   Repeat this step until the nodes in the list reflect how you want them to appear in the Model structure.

4. Click **Apply**.

29.8 Creating a Model Reference

For general information about References, see Chapter 4.

To create a Reference:

1. Open a Model for editing in the Structure area of the Workbench.
2. Click the icon in the **Create** column next to a Model or Component.

   After you create the Reference, this node is the Reference's parent in the Model structure.

3. Select **Create Model Reference**, then click **Continue**.
4. Select a Model from the list, then click **Finish**.

   The list includes all Models that appear in the Main area of the Repository, including the Model you are modifying.

29.8.1 Modifying a Model Reference

To modify a Model Reference:

1. Open a Model for editing in the Structure area of the Workbench.
2. In the same row as the Reference you want to modify, click the icon in the **Edit** column.
3. If you do not want the Reference to appear at runtime, deselect the **Display in User Interface** check box.
4. Optionally modify the Reference's Instantiability settings.

   For details, see Section 29.17.6, "Instances" on page 29-13.

5. Optionally modify the Reference's Effectivity.

   For details, see Section 29.17.12, "Effectivity" on page 29-14.

6. Click **Apply**.

29.9 Creating a Connector

For general information about Connectors, see Section 9.11 on page 9-6.

To create a Connector:

1. Open a Model for editing in the Structure area of the Workbench.
2. Click the icon in the **Create** column next to a Model or Component.
3. Select **Create Model Connector**, then click **Continue**.

4. Select a target Model from the list, then click **Finish**.

   The list includes all Models that appear in the Main area of the Repository, including the Model you are modifying.

5. Verify that the Model that is the target of the Connector will exist at runtime.

   If a Connector's target is not part of the Model you are editing, perform one of the following:

   - Create a Reference to the Connector's target Model
   - Create a Reference from the Connector's target Model to the Model you are currently editing (to do this, you must first open the target Model for editing)
   - Reference both the Connector's parent Model and the Connector's target Model from the same (parent) Model

   Creating Model References is described in Section 29.8 on page 29-5.

   Note: If the target Model does not exist in your Model's structure, the target Model cannot be instantiated at runtime. Therefore, no targets will be available when an end user clicks the **Choose Connection** button at runtime. For more information about Connectors and target Models, see Section 8.2 on page 8-3.

29.9.1 Modifying a Connector

To change a Connector's target, you must delete the Connector, recreate it, and then choose a different target Model.

To modify a Connector:

1. In the same row as the Connector you want to modify, click the icon in the **Edit** column.
2. If you do not want the Connector to appear at runtime, deselect the **Display in User Interface** box.
3. If connecting this Connector at runtime is not required to create a valid configuration, deselect the **Connection Required** box.
4. Optionally modify the Connector's Effectivity.

   For details about Effectivity, see Section 29.17.12 on page 29-14.
5. Click **Apply**.

29.10 Creating and Modifying Populators

For general information about Populators, see Chapter 10.

Perform the following to add a Populator to a node or edit an existing Populator:

1. Open a Model for editing.
2. Open the root node (if it is not an imported Model), a Component, or a Feature for editing.
3. In the node's details page, select the type of Item Master data you want the Populator to use when creating new Model structure from the **Source** list.

   All types of Item Master data are available in this list, regardless of the type of node you want the Populator to create. Select one of the following:

- Items
- Item Types
- Properties
- Property Values

4. Select the type of structure to create from the **Destination** list, then click **Go**.
5. In the Define Populator page, enter a unique **Name** for the Populator.
6. In the **Definition** section, specify the Item Master data you want the Populator to use.

   For details about the available choices and selection criteria, see Section 29.11, "The Define Populator Details Page" on page 29-7.

   The fields in this page are case-sensitive, so using the list of values to make a selection is recommended. If you do not enter the name of the item or Property exactly as it appears in the Item Master, Configurator Developer displays an error when you click **Update** to review the nodes that the Populator will create, or click **Apply** to save the Populator.
7. Click **Update** to view the effects of running the Populator before actually creating the new structure.

   The results appear in the **Preview Results** section.
8. If the results are not what you intended, modify the Populator's definition.
9. When you are satisfied with the Populator's definition, click **Apply** to return to node's details page.
10. Click **Apply**.

    This saves the Populator's definition on the selected node and executes the Populator (that is, it creates the new Model structure).

    If you do not want to save the Populator and do not want to create any new Model structure, click **Cancel**.

29.11 The Define Populator Details Page

The settings in the Define Populator page are determined by the type of Item Master data you select in the **Source** list in the node's details page. The field prompt in the Define Populator page changes based on the value of the **Source**. For example, when you choose Item Types, Items, or Properties, the Where *source value* prompt appears as **Where Item Type**, **Where Item is of type**, or **Where Property belongs to Item Type**, respectively. Enter criteria in the field next to the Where <*source value*> field by either typing text into the field or by clicking the flashlight icon (to select from a list of values).

You can create *all* nodes available in the Item Master under the selected node by entering the wildcard character (%). If you do this and are not satisfied with the results, you can remove the new structure by deleting the Populator after it runs. For details, see Section 29.13, "Deleting a Populator" on page 29-8.

If you select **Property Values** from the **Source** list, you must specify criteria differently. In this case, the Define Populator page displays two fields: **Where Item Type is** and **And Property is**. Click the flashlight icon to select an Item Type and Property.

Table 29–1 lists the available Item Master data types, the defining criteria for the Populator, and how you enter criteria based on the selected data type.

Table 29–1 Defining a Populator

Type of Item Master Data	Available Criteria	How you Specify Criteria
Item Types	begins with ends with contains matches	Type text into the field. Item Types are selected based on the criteria you enter.
Items	is of Type	Click the flashlight icon next to the field, then select from the list of Item Types.
	begins with endswith contains matches	Type text into the field. Item Types are selected depending on whether the Item Type name begins with, ends with, contains, or matches the text you enter.
Properties	Property belongs to Item Type	Click the flashlight icon next to the field, then select from the list of Item Types.
	begins with endswith contains matches	Type text into the field. Item Types are selected depending on whether the Item Type name begins with, ends with, contains, or matches the text you enter.
Property Value	Where Item Type is	Click the flashlight icon next to the field, then select from the list of Item Types.
	And Property is	Click the flashlight icon next to the field, then select from a list of the selected Item Type's Properties.

29.12 Repopulating Model Data

Repopulating updates a Model with data that has changed in the Item Master.

This procedure automatically runs all Populators defined within the Model that is open for editing.

For general information about Populators, see Chapter 10.

To repopulate a Model:

1. Open the Model for editing.
2. In the General area of the Workbench, click **Repopulate**.

29.13 Deleting a Populator

Deleting a Populator deletes all Model structure that was created using the Populator. To recreate any deleted Model structure, you must re-create the Populator, and then run it.

For general information about Populators, see Chapter 10.

To delete a Populator:

1. Open a Model, Component, or Feature for editing in the Structure area of the Workbench.

The Populator table lists each Populator defined for the selected node.

2. In the same row as the Populator you want to delete, click the icon in the **Delete** column.
3. Click **Yes** to acknowledge the confirmation message.

29.14 Adding Properties to a Model Node

For general information about Properties, see Chapter 5.

To add a Property to a node:

1. Open the Model for editing in the Structure area of the Workbench.
2. In the same row as the node you want to modify, click the icon in the **Edit** column.
3. In the node’s details page, click **Manage Properties**.
4. Select the Property you want to add, then click **Add to Selected List**.
5. In the Selected List table, optionally modify the Property’s value (see Note below).

Note: If you modify the Property’s value when adding it to a node, the change does not propagate to other nodes that share the Property. If you want the change to affect all nodes that share the same Property, refer to the procedure in Section 25.4.2 on page 25-6.

6. Click **Apply**.

29.15 Removing Properties from a Model Node

For general information about Properties, see Chapter 5.

The following procedure does not delete a Property from the CZ schema. Deleting Properties is described in Section 25.5.2, "Deleting Objects" on page 25-10.

To remove a Property’s association with a Model node:

1. Open the node for editing.
2. In the node’s details page, select the Property, and then click **Remove**.
3. Click **Apply**.

Note: You cannot remove a Property from a BOM node if the association was defined in Oracle Bills of Material (in other words, the Property and BOM node were imported into the CZ schema together). You can remove a Property from a BOM node only if the Property was assigned to the node in Configurator Developer.

29.16 Modifying Structure Node Properties

When editing a node, you can change the value of any Properties that were assigned to the node in Configurator Developer, even if the node is a BOM item or the Property was imported from Oracle Bills of Material. If a Property was imported with the BOM item you are editing, the Property’s value is read-only and cannot be modified in Configurator Developer.

A node's details page indicates whether a Property was imported from Oracle Bills of Material, or inherited (for example, by running a Populator).

To modify a Property assigned to a specific node:

1. Open the Model for editing in the Structure area of the Workbench.
2. In the same row as the node you want to modify, click the icon in the **Edit** column.
3. In the node's details page, modify the Property's **Value**, then click **Apply**.

29.17 Modifying Model Node Details

A node's details page displays detailed information about the node, such as its name, description, associated rules, effectivity, and so on. The following sections describe the node details page and indicate whether the information applies to only specific types of nodes, or all node types.

You can control which details appear in the Structure area of the Workbench by creating or modifying a View. For details, see Section 24.1.1, "Views" on page 24-2.

29.17.1 Name

This field displays the name of the selected node and is available for all nodes. When building Model structure, enter a name that will help you and others easily identify the node. The names of BOM nodes are imported with the BOM Model and cannot be modified in Configurator Developer. The names of imported BOM nodes are often part numbers.

You can use the same name for multiple nodes, since the CZ schema's internal ID for each node is unique. However, Model structure nodes must be unique within the same parent. For example, within a Component, you cannot create two Features with the same name.

If you create a node using a Populator, then the name is set automatically to the name of the corresponding Item in the Item Master. For more information, see Chapter 10.

Importing a BOM Model creates Items and Item Types in the Item Master. A profile option determines the default name of each Item Type. For more information, see the *Oracle Configurator Installation Guide*.

When you create Items in the Item Master manually, Oracle Configurator Developer assigns a default name based on the node's internal ID in the CZ schema. You can enter a more descriptive name to identify Items more easily.

A node's name and internal ID are included when you generate a Model Report. For details about generating a Model Report, see Section 28.1 on page 28-1.

Use settings in the Preferences page to control whether Configurator Developer displays nodes using their names or descriptions. For details, see Section 24.3.3 on page 24-8.

Note: Enter names that are unique, meaningful to the Oracle Configurator end user and, if possible, descriptive of the node's intended purpose. Names like **Response 1** and **Response 2** can easily cause confusion among Model developers as well as Oracle Configurator end users.

29.17.2 Description

Use this field to enter a description of any node that you create in Configurator Developer. Descriptions of imported BOM items are entered in Oracle Inventory, and are therefore read-only in Configurator Developer.

29.17.3 Display in User Interface

Deselect the Display in User Interface check box if you do not want Configurator Developer to generate a UI element for the selected node or any of its children. (In other words, you do not want the node to appear in a generated User Interface.)

You can override this setting when generating a new UI by selecting Show entire Model. Generating a new UI is described in Section 31.2 on page 31-1.

Some nodes might not appear at runtime because of their effective dates, or other display conditions that you define. For details about displaying nodes based on variable criteria, see Section 31.3.10, "Defining a Condition for Runtime Display and Behavior" on page 31-27.

For general information about effectivity, see Chapter 6.

29.17.4 Transient

Select the Transient check box to mark a node as transient. The meaning of marking a node as transient depends on the type of the node:

- If the node is a BOM Standard Item, it is a transient item. See Section 29.17.4.1, "Transient Items" on page 29-11.
- If the node is a Feature, it is a transient attribute. See Section 29.17.4.2, "Transient Attributes" on page 29-11.

This setting is relevant only for Models that allow reconfiguration of installed configuration instances. For more information about transient items and attributes, and about reconfiguring installed instances, see the *Oracle Telecommunications Service Ordering Process Guide*.

29.17.4.1 Transient Items

A transient item is a BOM Standard Item used to model a non-recurring service or fee, such as an initial installation fee. Transient items are omitted from the configuration session when a trackable instance of a service is reconfigured.

Transient items:

- Must be BOM nodes
- Must be non-trackable
- Cannot have children (since the children would also be dropped when an instance is reconfigured)

Therefore only BOM Standard Items (which cannot have children) can be transient items.

29.17.4.2 Transient Attributes

Transient attributes are conceptually similar to transient items. After a service is fulfilled, attributes that are transient are not restored during the reconfiguration of the service.

A transient attribute is a Feature used to model a non-recurring attribute of a Model node. The value of the Feature is what is regarded as transient. Values of transient attributes (that is, values of Features flagged as Transient) are omitted from the configuration session when a trackable instance of a service is reconfigured.

Attribute values are assigned by implementing the IB Attribute Configurator Extension. See the *Oracle Telecommunications Service Ordering Process Guide* for details.

Transient attributes:

- Must be Features (Integer, Decimal, Boolean, Text or Option Feature)
- Cannot be Options of an Option Feature
- Cannot be Resources or Totals

29.17.5 Definition

Use the settings in this section to enter basic information about a node. The settings and information that appear here depend on the node's type, and whether it is an imported BOM node.

For BOM nodes, the information in the Definition section is read-only; this is because these settings are defined in either Oracle Inventory or Oracle Bills of Material. The BOM Item Type indicates whether the selected item is a BOM Model, BOM Option Class, or a BOM Standard Item. For details about the Minimum Quantity, Maximum Quantity, and Default Quantity values, see Section 11.3, "Imported BOM Rules" on page 11-3.

In the details page for a BOM item, the Definition section indicates whether:

- Optional children are mutually exclusive (see Section 11.3, "Imported BOM Rules" on page 11-3)
- The item is required when its parent is selected (see Section 11.3, "Imported BOM Rules" on page 11-3)
- Decimal quantities are allowed (see Section 3.3.5.1, "Decimal Quantities and BOM Items" on page 3-4)
- The item is trackable. This setting is relevant only for Models that allow reconfiguration of installed instances.

  For more information, refer to the *Oracle Telecommunications Service Ordering Process Guide*.

If the selected node is an Integer Feature, Decimal Feature, or Option Feature, refer to the following sections for information about the Minimum Selections, Maximum Selections, and Enable Option Quantities settings:

- Section 9.7.2, "Integer Features" on page 9-3
- Section 9.7.3, "Decimal Features" on page 9-4
- Section 9.7.1, "Option Features" on page 9-3

For details about the Initial Value setting, see Section 9.12, "Initial Values" on page 9-6.

29.17.5.1 Connection Required Setting

If the selected node is a Connector, the Connection Required box indicates whether an end user must connect the selected node and its target at runtime to create a valid configuration.

For more information, see Chapter 8, "Connectivity".

29.17.6 Instances

If the selected node is a configurable component (in other words, a Component or a Model Reference node), use the Instantiability settings to define how many instances of the node may exist, or be created, at runtime. Select from the following:

- **Required Single Instance**: Select this setting to allow one and only one instance of the selected component in the configuration. In other words, one instance of the component is available when the configuration session begins, and the end user cannot create any additional instances.
- **Optional Single Instance**: Select this setting if you do not want any instances of the selected component to be available when the configuration session begins, but want end users to be able to add one instance of the component.
- **Multiple or Variable Instances**: Select this setting to specify how many instances of the component exist when the configuration session begins, and how many instances the end user can add.

  To make a selection mandatory, set the Initial Minimum to 1 or more. To make a selection optional, set the Initial Minimum to 0 (zero). For example, if a component's Initial Minimum is 1 and its Initial Maximum is 10, at least 1 instance of the component must exist in the configuration, but the end user can create up to 9 additional instances.

  Note: If you modify the Initial Minimum or Initial Maximum values, refresh the User Interface to be sure that the new setting is used at runtime.

If your Model has one or more Numeric Rules that dynamically changes how many of this component's instances are allowed based on end user selections, you must select Multiple or Variable Instances for the rule to function. For details about this type of rule, see Example 13–2 on page 13-5.

For more information about instantiating components at runtime, and the effect that changing these values has on a saved configuration, see Chapter 7.

29.17.7 Properties

Properties provide additional information about a node (for example, weight, diameter, or voltage). See Section 29.14, "Adding Properties to a Model Node" on page 29-9.

For more information, see Chapter 5, "Properties".

29.17.8 Populators

This section lists any Populators associated with the selected node and enables you to edit or delete an existing Populator, or define a new Populator.

For more information, see Chapter 10, "Using Populators".

29.17.9 Associated Rules

This read-only section lists any rules in which the selected node is a participant. You may want to view this information before modifying or deleting a node because such changes can affect how a rule performs or cause it to become invalid.

29.17.10 Associated UI Nodes

If a UI has been generated for the Model, this read-only section lists information about how the selected node appears in the UI. This information includes the UI element name, its type, and the UI Page in which the selected element appears (the Enclosing Page).

For more information, see:

- Chapter 19, "Displaying the Model"
- Chapter 21, "User Interface Structure and Design"
- Chapter 29, "Structure Area of the Workbench"

29.17.11 Violation Message

Use this section to define a violation message that appears at runtime when a Resource is over-consumed. The default message is "The resource *resource name* is over-consumed."

For more information, see:

- Section 9.9, "Totals and Resources" on page 9-5
- Section 29.5, "Creating a Total or Resource" on page 29-3

For details about defining a violation message for a rule, see Section 30.19.2 on page 30-18.

29.17.12 Effectivity

These settings control whether a Model node appears in a runtime UI or when unit testing a configuration model in the Model Debugger. By default, nodes that you create in Configurator Developer are set to Always Effective. You cannot modify the effectivity settings for imported BOM nodes.

For general information about effectivity, see Chapter 6.

The following procedures assume you have a Model open for editing and are working in the Structure area of the Workbench.

To specify a range of dates for a non-BOM node:

1. Open the node for editing.
2. In the **Effectivity** section of the node's details page, select **Choose Explicit Dates** from the **Action** list, and then click **Go**.
3. Select **No Start Date** or **No End Date**, or specify a From and To date and time.

   You can specify a very wide range of dates when entering a start and end date, but this range is limited. For more information, see the *Oracle Configurator Implementation Guide*.
4. Click **Apply**.

To assign an Effectivity Set to a non-BOM node:

1. Open the node for editing.
2. In the **Effectivity** section of the node's details page, select **Choose Effectivity Set** from the **Action** list, then click **Go**.

   The Select Effectivity Set page lists all Effectivity Sets in the CZ schema.

3. Select an Effectivity Set, and then click **Apply**.

To assign a Usage to a node:

1. Open the node for editing.
2. In the **Effectivity** section of the node's details page, click **Select Usage Exceptions**.
3. Select the Usage(s) for which the node is not effective, and then click **Apply**.
4. In the node's details page, click **Apply**.

For more information about Usages, see Chapter 6.

29.17.13 Notes

Use this field to enter any additional information about the selected node, such as its purpose or when it was created. This field accepts a maximum of 2000 characters.

29.18 The Structure Area of the Workbench Actions List

The Actions list in the Structure area of the Workbench includes Copy, Move, Delete and Rename.

You can copy and move nodes in the Structure area of the Workbench with the following limitations:

- Option nodes can be copied or moved only to a Feature node.
- Feature nodes can be copied or moved only to a Component node.
- Component nodes can be copied or moved only to another Component or Model node.
- Total and Resource nodes can be copied or moved only to a Component node.
- When you move a node that has one or more Populators defined, the Populator remains with the node after the move. However, when you copy a node that has one or more Populators, the Populators are not retained.

The procedures for performing the Copy, Move, Delete and Rename actions are the same for Model structure nodes and objects in the Main area of the Repository. For details, see:

- Section 25.5.1, "Moving and Copying Objects" on page 25-9

  See also Section 29.19, "Copying Rules Associated with a Component" on page 29-15.
- Section 25.5.2, "Deleting Objects" on page 25-10
- Section 25.5.3, "Renaming Objects and Modifying Descriptions" on page 25-10

29.19 Copying Rules Associated with a Component

When you copy nodes or rules, you must specify a destination for the copied objects. When you copy a Component node, an additional setting is available. This setting is called Copy Rules Associated with Components. When you select this setting:

- Configurator Developer copies only the rules whose participants are entirely contained within the substructure of the Component you are copying, and places the copy of each rule in the corresponding rule Folder.

- If any of the nodes in a rule are not within the structure of the Component you are copying, then that rule is not copied.

You may want to select this setting if, for example, you want to constrain the copied structure the same way as the source structure, but do not want to recreate all of the rules from scratch.

30 Rules Area of the Workbench

This chapter describes the tasks you can perform in the Rules area of the Workbench.

This chapter includes the following sections:

30.1 Introduction

Use the Rules area of the Workbench to define rules and modify existing rules. To modify how Model structure nodes are displayed in this area of the Workbench, see Section 24.3.3, "Preferences" on page 24-8.

When you create a rule, Oracle Configurator Developer creates a new node in the rule Folder you specified. You can create as many Folders as you need, and a Folder can contain any type of rule. Use the Move and Copy actions to move or copy a rule from one Folder to another. Creating rule Folders is described in Section 30.12 on page 30-15.

When you need to define more complex configuration rules, create a Statement Rule. For details, see Section 30.9 on page 30-9.

You can also extend the abilities of Configurator Developer and define custom behavior by creating **Configurator Extensions**. Configurator Extensions enable you to implement custom runtime rules, access information outside the configuration model, perform engineering calculations and integrate with an alternative user interface. Configurator Extensions are described in more detail in Chapter 17.

For general information about rules, see Chapter 11.

30.2 Defining Rules

Follow these general steps to define any type of configuration rule:

1. Navigate to the Rules area of the Workbench.
2. In the same row as either the predefined *Model Name* Rules Folder, or a user-defined Folder, click the icon in the **Create** column.
3. Select the type of rule you want to create, then click **Continue**.
4. Enter the following for the rule:
   - "Name" on page 29-10
   - "Description" on page 29-11
   - "Definition" on page 30-18
   - "Violation Message" on page 30-18
   - "Unsatisfied Message" on page 30-19 (available only for Logic and Numeric Rules)
   - "Effectivity" on page 6-1
   - "Notes" on page 29-15
5. When the rule's definition is complete, optionally generate logic and test the rule in the Model Debugger, or a runtime UI. For details, see:
   - Section 28.7, "Logic Generation Status" on page 28-5
   - Chapter 22, "Testing and Debugging"

Refer to the following sections for information on defining a specific type of rule:

- Section 30.3, "Defining Logic Rules" on page 30-3
- Section 30.4, "Defining Numeric Rules" on page 30-3
- Section 30.5, "Defining Comparison Rules" on page 30-4
- Section 30.9, "Defining Statement Rules" on page 30-9
- Section 30.7, "Defining Explicit Compatibility Rules" on page 30-7
- Section 30.6, "Defining Property-based Compatibility Rules" on page 30-6
- Section 30.8, "Defining Design Charts" on page 30-8
- Section 30.10, "Creating a Rule Sequence" on page 30-9
- Section 30.11, "Creating a Configurator Extension Rule" on page 30-10

30.3 Defining Logic Rules

For an overview of Logic Rules, see Chapter 12.

The following procedure assumes you have a Model open for editing.

To define a Logic Rule:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Logic**, then click **Continue**.
3. Enter a **Name** and optionally a **Description** of the rule.
4. Under **Operand 1**, click **Choose Nodes**.
5. Select the node(s) to add to the rule, and then click either **Choose Properties** or **Apply**.

   Click **Apply** to add the node(s) and return to the rule details page.

   Click **Choose Properties** if you want to specify a System Property to use in the rule. After selecting a Property for each node, click **Finish**.

   For more information, see Appendix C, "Rules, Node Types, and System Properties" on page C-1.
6. Select a Condition of **Any True** or **All True**.

   For details, see Section 12.3, "Using AllTrue and AnyTrue" on page 12-5.
7. Choose one of the following rule operands: **Implies**, **Excludes**, **Requires**, **Negates**, or **Defaults**.

   For a description of each operand, see Section 12.1, "Logical Relationships" on page 12-1.
8. Use the procedure described above to select rule participants for **Operand 2**.
9. If you need to define a more complex expression, click **Convert to Statement Rule**.

   For details, see Section 30.9, "Defining Statement Rules" on page 30-9.
10. Optionally define the following: **Violation Message**, **Unsatisfied Message**, **Effectivity**, and **Notes**.

    For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
11. Click **Apply** or **Apply and Create Another**.

30.4 Defining Numeric Rules

This section describes how to define a simple Numeric Rule. To define more complex Numeric Rules, create a Statement Rule. For details, see Chapter 16.

For an overview of Numeric Rules, see Chapter 13.

The following procedure assumes you have a Model open for editing.

To define a Numeric Rule:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Numeric**, then click **Continue**.

3. Enter a **Name** and optionally a **Description** of the rule.
4. Under **Operand 1**, click **Choose Nodes**.
5. Select the node(s) to add to the rule, and then click either **Choose Properties** or **Apply**.

   Click **Apply** to add the node and return to the rule's details page.

   Click **Choose Properties** if you want to specify a System Property to use in the rule. After selecting a Property for each node, click **Finish**.

   For more information, see Appendix C, "Rules, Node Types, and System Properties" on page C-1.
6. Indicate whether the result of Operand 1 will **Contribute** to or **Consume** from the participant in Operand 2 (you select this node in the next step).
7. Repeat the previous steps to select a single node under **Operand 2**.
8. Enter either an integer or decimal value as the **Quantity Multiplier**, or accept the default value of 1.

   For example, you enter a Quantity Multiplier of 5. When the rule propagates at runtime, the value of the node under Operand 1 is 10. Therefore, a value of 50 will be contributed or consumed from Operand 2.
9. If the node in **Operand 2** is not an Integer Feature, optionally select one of the following **Integer Conversion** methods:
   - Accept the default value of **None** if the node in Operand 2 is an Integer Feature, or if the node has a data type of Decimal and you do not want to convert the result to an integer.
   - Select **Round** to convert the result to the nearest integer.
   - Select **Floor** to convert the result to the next lower integer.
   - Select **Truncate** to convert the result to an integer by removing any numbers after the decimal. For example, 4.5687 becomes 4, and 12.879 becomes 12.

     Floor and Truncate produce the same result when applied to a positive value. The difference between the two methods is apparent when the value is a negative number. For example, when the value is 4.3, it is converted to 4 regardless of whether you select Floor or Truncate. However, if the value is -4.3, the result is –5 when you select Floor but –4 when you select Truncate.
   - Select **Ceiling** to convert the result of Operand 1 to the next higher integer.
10. If you need to define a more complex expression, click **Convert to Statement Rule**.

    For details, see Section 30.9, "Defining Statement Rules" on page 30-9.
11. Optionally define the following: **Violation Message**, **Unsatisfied Message**, **Effectivity**, and **Notes**.

    For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
12. Click **Apply** or **Apply and Create Another**.

30.5 Defining Comparison Rules

For an overview of Comparison Rules, see Section 15.1 on page 15-1.

The following procedure assumes you have a Model open for editing.

To define a Comparison Rule:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Comparison**, then click **Continue**.
3. Enter a **Name** and optionally a **Description** of the rule.
4. Under Operand 1, click **Choose Node**.
5. Select the node to add to the rule, and then click either **Choose Properties** or **Apply**.

   Click **Apply**.

   Click **Choose Properties** if you want to specify a System Property to use in the rule. After selecting a Property for each node, click **Finish**.

   For more information, see Appendix C, "Rules, Node Types, and System Properties" on page C-1.
6. Select one of the following **Conditions** from the list:
   - = (Equals)
   - < > (Not equals)
   - > (greater than)
   - >= (greater than or equal to)
   - <= (less than or equal to)
   - < (less than)
7. Choose one of the following operands:
   - **Constant:** If you select this option, enter a positive numeric value in the field provided.
   - **Model Node**: If you select this option, click **Choose Node** and then select a Model node.
8. Indicate whether the result of Operand 1 **Implies**, **Excludes**, **Requires**, **Negates**, or **Defaults** the node in Operand 2.

   For details about each type of relation, see Section 12.1, "Logical Relationships" on page 12-1.
9. Click **Choose Nodes** to add one or more nodes to Operand 2.
10. Select a Condition of **AnyTrue** or **AllTrue**.

    For details, see Section 12.3, "Using AllTrue and AnyTrue" on page 12-5.
11. If you need to define a more complex expression, click **Convert to Statement Rule**.

    For details, see Section 30.9, "Defining Statement Rules" on page 30-9.
12. Optionally specify a **Violation Message**, **Unsatisfied Message**, **Effectivity**, and **Notes** for the rule.

    For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
13. Click **Apply** or **Apply and Create Another**.

30.6 Defining Property-based Compatibility Rules

For an overview of Property-based Compatibility rules, see Section 15.2.2 on page 15-3.

The following procedure assumes you have a Model open for editing.

To define a Property-based Compatibility Rule:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Property-based Compatibility**, and then click **Continue**.
3. Enter a rule **Name** and optionally a **Description**.
4. Under **Operand 1**, click **Choose Node**.
5. Select an Option Feature or a BOM Option Class node, and then click **Apply**.
6. Select a **Child Property** from the list.
7. Select one of the following comparison operators from the list:
   - `BeginsWith`
   - `=` (equals)
   - `EndsWith`
   - `<>` (not equals)
   - `Contains`
   - `>` (greater than)
   - `Like`
   - `>=` (greater than or equal to)
   - `<` (less than)
   - `<=` (less than or equal to)

   See Section 30.6.1 on page 30-7 for more information about the equals (=), Contains, and Like operators.

   > **Note:** You can use any of these operators when comparing either numeric Properties or text Properties.

8. Under **Operand 2**, click **Choose Node**.
9. Select a Feature or BOM Option Class node from the Model structure, and then click **Apply**.
10. Select a **Child Property** from the list.
11. Optionally define the following: **Violation Message**, **Effectivity**, and **Notes**.

    For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
12. If you need to add another Property comparison to the rule or define a more complex expression, click **Convert to Statement Rule**.

    For details, see Section 30.9, "Defining Statement Rules" on page 30-9.
13. Click **Apply** or **Apply and Create Another**.

30.6.1 Equals, Contains, and Like Operators

These operators are used when comparing Properties that consist of text, rather than numeric values. For example, a Property for the BOM Standard Item Copper Pipe is Diameter.

Properties may contain the percent sign (%) and underscore (_) characters. By using these characters, the equals (**=**), Contains, and Like operators enable you to apply the Property-based Compatibility Rule to only specific options. For example, when using the equals operator (=), the rule checks whether the Property in Operand 2 contains the percent sign or underscore character itself.

When used with the Contains or Like operators, the percent sign and underscore characters act as "wildcards." The percent sign can represent zero or more characters, while the underscore represents a single character. For example, "Dia%" will find a match for any word that begins with "Dia", while "_eight" will find a match for both "Height" and "Weight".

For example, the rule `AddressFeature.address Contains "De"` is the same as `AddressFeature.address Like %De%`". Both rules find a match if the Property in Operand 1 contains "De".

The equals operator does *not* use the percent sign or underscore characters as wildcards. So, the rule `AddressFeature.address = "Diam%"` does not find a match if the Property in Operand 2 is Diameter; it finds a match only if the Property is "Diam%".

30.7 Defining Explicit Compatibility Rules

For an overview of Explicit Compatibility rules, see Section 15.2.3 on page 15-4.

The following procedure assumes you have a Model open for editing.

To define an Explicit Compatibility Rule:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Explicit Compatibility**, then click **Continue**.
3. Enter a **Name** and optionally a **Description** of the rule.
4. Click **Choose Nodes** to select the rule's participants.
5. Select one or more Option Features or BOM Option Classes, and then click **Apply**.

   The Compatible Combinations table displays each node you selected in a separate column.
6. Use the cells of the table to specify compatible combinations of Feature Options and BOM Standard Items.
7. Click **Add Another 5 Rows** to define additional combinations.

   The same Option or BOM Standard Item can appear in a column as many times as necessary to define the compatibility table you need.
8. Optionally specify a **Violation Message**, **Unsatisfied Message**, **Effectivity**, and **Notes** for the rule.

   For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
9. Click **Apply** or **Apply and Create Another**.

30.8 Defining Design Charts

For an overview of Design Charts, see Chapter 14.

To define a Design Chart:

1. Go to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Design Chart**, then click **Continue**.
3. Enter a **Name** and optionally a **Description** of the Design Chart.
4. Click **Choose Primary Feature**, and then select an Option Feature or BOM Option Class node.
5. Click **Apply**.
6. Click **Add Defining Feature**, and then select one or more Option Features or BOM Option Class nodes.

   Note: There is no limit to how many Defining Features you can include in a Design Chart. However, Defining Features should always have a Minimum Selections = 0 or 1, and Maximum Selections = 1. This is because allowing the end user to choose more than one option from a Defining Feature can make it difficult to map choices to a particular Primary Feature selection.

7. Click **Apply**. A table row is automatically populated for each of the Defining Feature's Options.

   Configurator Developer populates the **Design Chart** table with a column for each of the Primary Feature's options and a row for each of the Defining Feature's options.

   A Primary Feature must be defined before the table is saved and you can activate the compatibility cells.

8. Click **Add Optional Feature** and then select one or more Option Features or BOM Option Class nodes. There is no limit to how many Optional Features you can include in a Design Chart.
9. Click **Apply**.

   Configurator Developer populates a table row for each of the Optional Feature's options.

10. When the table contains all of the Defining and Optional Features, select the check box in the appropriate table cells to specify compatibilities between the Defining Feature options and the options of the Primary and Optional Features.

    An Optional Feature Option can be compatible with multiple Primary Feature Options, and multiple Options of a given Optional Feature can be compatible with a given Primary Feature Option.

11. To remove an Optional Feature and all of its options from the table, click the icon in the **Delete** column.

    You cannot delete individual Options from a Design Chart.

12. Click **Validate Chart** to ensure the rule's definition is valid.

If a message indicates that the rule's definition is invalid, modify its definition and then click **Validate Chart** again. Repeat this step until the rule's definition is valid.

13. Optionally specify a **Violation Message**, **Effectivity**, and **Notes** for the rule.

    For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.

14. Click **Apply** or **Apply and Create Another**.

30.9 Defining Statement Rules

For general information about this type of rule, see Chapter 16, "Statement Rules".

The following procedure assumes you are have either performed the initial steps to create a new Statement Rule, or are converting a Logic, Numeric, Comparison, or Property-based Compatibility Rule into a Statement Rule. In other words, you are viewing a Statement Rule's details page.

To define a Statement Rule:

1. Enter a **Name** and **Description** of the rule.
2. Enter the rule's **Definition** using the Constraint Definition Language (CDL)

   You can also add Model structure nodes as rule participants by clicking **Choose Nodes**. When you do this, the name of the node appears at the end of the rule's definition.

   For details about using CDL, as well as required syntax and validation criteria, see the *Oracle Configurator Constraint Definition Language Guide*.
3. Click **Validate Rule Text** to ensure the rule's syntax is correct.
4. Optionally define the following: **Violation Message**, **Unsatisfied Message**, **Effectivity**, and **Notes**.

   For more information about these settings, see Section 30.19, "Modifying Rule Details" on page 30-18.
5. Click **Apply** or **Apply and Create Another**.

30.10 Creating a Rule Sequence

For general information about Rule Sequences, see Chapter 18.

All rule types except Configurator Extensions can participate in a Rule Sequence. However, a Rule Sequence cannot contain another Rule Sequence.

To create a Rule Sequence:

1. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
2. Choose a rule type of **Rule Sequence**, then click **Continue**.
3. Enter a **Name** and optionally a **Description**, and then click **Apply**.
4. Add rules to the Rule Sequence.

   There are two ways to add rules to a Rule Sequence. You can either create a new rule within the Rule Sequence, or add an existing rule to the Rule Sequence.

   To create a new rule within a Rule Sequence:

   a. In the same row as the Rule Sequence, click the icon in the **Create** column.

b. Select the type of rule to create, then click **Continue**.

c. Define the rule.

   For details, see Section 30.2, "Defining Rules" on page 30-2.

To add an existing rule to a Rule Sequence:

a. Select a rule.

   You can move only one rule at a time into a Rule Sequence.

b. Select **Move** from the Actions list, and then click **Go**.

c. Select the Rule Sequence, and then click **Apply**. You might need to expand one or more rule Folders to be able to select the Rule Sequence.

   Configurator Developer places the rule at the end of the sequence and sets it to Never Effective.

d. To modify the rule's effective dates, click the icon in the **Edit** column.

   For additional steps, see Section 29.17.12 on page 29-14.

e. Click **Apply**.

30.11 Creating a Configurator Extension Rule

For an overview of Configurator Extensions, see Chapter 17, "Configurator Extensions".

Before you can bind any Java methods to your Model, you must create a Configurator Extension Rule in which to define the bindings. The Configurator Extension Rule specifies the node and the Java class for which the binding will be defined.

Configurator Extension Rules have many of the same settings as other rules, and the procedure for defining them is similar. For general information about defining configuration rules, see Section 30.2, "Defining Rules" on page 30-2.

To create a Configurator Extension Rule:

1. In the Main area of the Repository, locate the Model containing the node that you that you want to associate with a Configurator Extension and click the icon in the **Edit** column in the same row as the Model.
2. Navigate to the Rules area of the Workbench, and then click the icon in the **Create** column.
3. Choose a rule type of **Configurator Extension**, then click **Continue**.
4. Enter the **Name** and **Description** of the rule.
5. Optionally, you can click **Disable** to prevent the rule from operating.

   Disabling a Configurator Extension Rule can be useful when the Java class for your Configurator Extension is not yet ready for testing. Otherwise, if you leave the rule unfinished, it will cause a runtime error when you test the Model.
6. Optionally, you can use the **Effectivity** section to set the effectivity of the rule.

   See Section 29.17.12, "Effectivity" on page 29-14 for details.
7. Configurator Extension Rules do not have settings for a Violation or Unsatisfied Message. These messages should be generated by the Configurator Extension itself.

8. Associate the Configurator Extension Rule with a node in your Model. See Section 30.11.1, "Associating a Node" on page 30-11.

30.11.1 Associating a Node

A Configurator Extension must be associated with a node in your Model. This node determines the event binding scope, which is explained in Section 17.4, "Events" on page 17-4.

1. In the **Definition** section of the rule, click **Choose Node**.
2. On the Choose Node page, select the node that you want to associate with the Configurator Extension.

   You can associate Configurator Extensions with any type of node. The node must belong to the Model that contains your Configurator Extension Rule.
3. Click **Apply.**

   You are returned to the Configurator Extension Rule page. In the Definition section, the name and path of the node you chose is displayed, identified by the label **Model Node**.
4. Now you can choose a Java class for the rule. See Section 30.11.2, "Choosing the Java Class" on page 30-11.

30.11.2 Choosing the Java Class

A Configurator Extension Rule must specify the Java class that contains the method that implements the functionality for your Configurator Extension.

To choose a Java class for a Configurator Extension Rule:

1. In the Definition of the rule, click **Choose Class**.
2. On the Choose Class page, a table lists the Java classes that are available to the Configurator Extension Rule for use in a binding.

   This list of bindable classes contains only the classes that are in the Archive Path of the current Model. The list of bindable classes consolidates *all* of the classes in *all* of the Configurator Extension Archives that are in the Archive Path of the current Model.

   Note: Do not confuse the list of bindable classes for a Model, described here, with the list of classes in a Configurator Extension Archive. The latter is described in Section 25.4.4.1, "Viewing the Classes in an Archive" on page 25-8.

   If a class with the same name exists in more than one Configurator Extension Archive, Oracle Configurator loads the class that occurs first in the Archive Path.

   For details on Configurator Extension Archives, see Section 17.3, "Configurator Extension Archives" on page 17-3. For details on how to set the Archive Path, see Section 28.10.2, "Adding Archives to a Model's Archive Path" on page 28-7.
3. Expand the table to display the names of the Java classes.

   - Each Java class in the Archive is represented by a row in the table.
   - The package hierarchy of the bindable classes, if any, is represented by the hierarchy of the table. Each package is represented by a row in the table that

includes a **Focus** control and a hide/show toggle. Click the hide/show toggle to view the next level of the package. To view the complete hierarchy of the archive, click **Expand All**. To narrow the focus of the view to a particular package, click its **Focus** control. To expand the focus again, click on a package name in locator links in the table heading.

- If the archive also contains Java source files for the classes, their names are shown too, but they are not enabled for selection.

4. Select the class that you want to bind to the Model node that you chose.
5. When you have chosen a class, click **Apply**.

   You are returned to the Configurator Extension Rule page. In the Definition, the name of the class you chose is displayed, identified by the label **Java Class**.

   The class name is editable, so that you can enter the fully qualified name of a class that is in the class path of the host application. See Section 17.3.1, "The Archive Path" on page 17-3 for background.
6. If the **Model Node** that you chose is one that can be instantiated multiple times, then you must choose the instantiation scope for the Configurator Extension. Do this by choosing one of the options for **Java Class Instantiation**, which are described in the following table. See Chapter 17, "Configurator Extensions" for background.

Option	Meaning
With Model Node Instance	The Configurator Extension is instantiated with each separate instance of the Model node.
With Model Node Instance Set	The Configurator Extension is instantiated once, with the entire set of instances of the Model node.

   If the **Model Node** that you chose cannot be instantiated multiple times, then no choice of instantiation scope is available. The Configurator Extension is instantiated at runtime with each separate instance of the Model node.
7. Now you can create an event binding for the rule. See Section 30.11.3, "Creating Event Bindings" on page 30-12.

30.11.3 Creating Event Bindings

Once you have chosen a Java class from for the Configurator Extension Rule, you create one or more event bindings for the rule. See Section 17.4, "Events" on page 17-4 for background.

Creating an event binding consists of binding an event to a Java class method. The available events are seeded or custom. The available methods are contained in the Java classes accessed by the Model's Archive Path.

To create one or more event bindings for a Configurator Extension Rule:

1. In the Definition of the rule, click **Create Binding**.
2. On the Add Event Binding page, the **Event** control lists the events to which you can bind a method of a Java class.

   Choose the Event to which you want to bind the execution of the Java method.

   See Table 17–2, " Predefined Events for Binding" on page 17-6 for a description of the available events.

3. If the event that you choose is a command event (`onCommand`), then the **Command Name** field appears.

   In this field, type the character string that is the name of the command that you wrote your Configurator Extension's Java method to handle. Do not enclose the string in quotation marks. The string can contain spaces.

4. The **Event Scope** control lists the event binding scopes that can be used to bind the event that you chose. Event binding scopes are described in Table 17–1, " Event Binding Scopes" on page 17-5. The scopes that apply to each event are listed in Table 17–2, " Predefined Events for Binding" on page 17-6.

   Set the event scope for this binding, by choosing one of the options for Event Scope. The list of available scopes is restricted to those that apply to the chosen event.

5. A table lists the public Java methods contained in the class that you chose (as described in Section 30.11.2, "Choosing the Java Class" on page 30-11). Each method is displayed with its parameters and their data types, as defined in the Java class. The arguments are automatically assigned names indicating their order in the method's parameter list, such as `Arg1`, `Arg2`, and so on.

   Select the Java method that is appropriate for the Configurator Extension functionality that you are defining.

   Methods selected for Configurator Extensions cannot have names longer than 30 characters. If you want to use a method with a longer name, then the Java class must be modified and then included in a fresh version of your Configurator Extension Archive.

6. Click **Continue**.

   Now you can proceed to bind arguments to the parameters of the Java method. To do this, see Section 30.11.4 on page 30-13.

Note: After you have created a binding, you cannot modify its attributes. However, you can delete the binding and create another, to set the attributes as you require.

30.11.4 Binding Arguments to Parameters

Once you have selected an event and a Java method, you bind each of the parameters of the method to arguments related to your configuration model. See Section 17.5, "Argument Binding" on page 17-9 for background.

On the Bind Method Arguments page, each argument of the method is represented by a row in the **Argument Bindings** table. The columns of this table are described in Table 30–1 on page 30-13. Configurator Developer generates this list of arguments by performing a Java introspection of the class, which is contained in a Configurator Extension Archive. See Section 17.3, "Configurator Extension Archives" on page 17-3 and Section 30.11.2, "Choosing the Java Class" on page 30-11 for background.

Table 30–1 Elements of an Argument Binding

Column	Meaning
Argument Type	The Java type of the argument (showing the package and class name), as specified in the definition of the method. Generated by introspection of the class.

Table 30–1 (Cont.) Elements of an Argument Binding

Column	Meaning
Argument Name	The name of the Java argument specified in the definition of the method.
Argument Specification	The type of the argument being bound to the method parameter. These types are described in Table 17–3, " Parameter Types for Argument Specification" on page 17-9.
Binding	The value that you are binding to the argument. The type of field presented here is determined by the type of Argument Specification.
Select Node or Property	If the Argument Specification is Model Node or Property, this control opens the Choose Node pages so that you can select one.

To bind an argument to a method parameter:

1. Locate the desired argument in the Argument Bindings table. Each argument is identified by its Java type (in the **Argument Type** column) and its name (in the **Argument Name** column). The arguments have names that are automatically assigned to indicate their order in the method's parameter list, such as `Arg1`, `Arg2`, and so on.
2. In the **Argument Specification** column, select the type of argument that you are binding to the parameter, from the following list:
   - System Parameter
   - Event Parameter
   - Model Node or Property
   - Literal

   These types are described in Table 17–3, " Parameter Types for Argument Specification" on page 17-9.
3. In the **Binding** column, apply an option that applies to the type of Argument Specification you chose.

Table 30–2 Binding Choices for Argument Specification

For this Argument Specification ...	Choose This Type of Binding ...
System Parameter	Choose a system parameter from the options displayed. For descriptions, see Table 17–3, " Parameter Types for Argument Specification" on page 17-9.
Event Parameter	Choose an event parameter. Only the event parameters for the selected event are displayed. For descriptions, see Table 17–2, " Predefined Events for Binding" on page 17-6.
Model Node or Property	Click the **Select Node** control for this argument. On the resulting Choose Node page, expand the Model to locate the desired node. Select the desired node, then click **Apply**, which returns you to the Add Event Binding page.
Literal	Type an unquoted character string.

The Java types of the parameters of your method must agree with the types of Model entities that are eligible for event binding. For a list of the Java classes that you can use in event bindings, see the *Oracle Configurator Extensions and Interface Object Developer's Guide*.

4. Repeat this process for each argument in the table. The order in which you bind the arguments is not significant.
5. Click **Finish** to complete the binding of the arguments. You can leave some arguments unbound, but doing so will cause errors when you generate logic, and at runtime.
6. You are returned to the Configurator Extension Rule page. Click **Apply** to finish your definition of this Configurator Extension Rule.

30.11.5 Generating Logic for Configurator Extensions

As with other configuration rules, after you create or modify a Configurator Extension Rule, you must generate logic before testing or using your configuration model. See Section 30.2, "Defining Rules" on page 30-2 and Section 28.7, "Logic Generation Status" on page 28-5.

There are some special considerations when generating logic for Configurator Extensions:

- If some part of the definition of a Configurator Extension Rule is omitted or invalid, then Configurator Developer produces warnings when you generate logic for the Model, and marks the Rule as invalid. During a configuration session, the runtime Oracle Configurator ignores this Configurator Extension.
- The logic generation process does not examine the contents of Configurator Extension Archives. If you modify a Java class in a way that affects the definition of the Configurator Extension Rules that use that class (for example, to add a parameter to a method) and upload the changed Archive containing the class, then you must make corresponding changes to the affected Configurator Extension Rules. When you generate logic, Configurator Developer is not able to warn you about differences between the Java class and Configurator Extension Rules. During a configuration session, the Configurator Extension may fail.

30.12 Creating a Rule Folder

For details about rule Folders, see Section 11.2.2 on page 11-2.

To create a rule Folder:

1. Open a Model for editing, and then navigate to the Rules area of the Workbench.
2. In the same row as the Configuration Rules Folder or any user-defined rule Folder, click the icon in the **Create** column.
3. Select **Folder**, then click **Continue**.
4. Enter a **Name** and optionally a **Description** of the Folder.
5. Click **Apply** or **Apply and Create Another**.

30.13 Deleting a Rule or Rule Folder

You can delete a rule at any time as long as it resides within the Model that is open for editing (in other words, it cannot belong to a referenced Model). Deleting a Folder deletes all of the rules it contains

The following procedure assumes you have a Model open for editing.

To delete a rule or rule Folder:

1. Navigate to the Rules area of the Workbench and locate the rule or Folder you want to delete.
2. Select the rule or Folder, and then select **Delete** from the **Actions** list.
3. Click **Yes** to confirm the action.

30.14 The Rules Area of the Workbench Actions List

The Actions list in the Rules area of the Workbench includes Copy, Move, Delete, Rename, Enable, and Disable. The procedures for copying, moving, deleting, and renaming rules are the same as for objects in the Main area of the Repository.

For details, see:

- Section 25.5.1, "Moving and Copying Objects" on page 25-9
- Section 25.5.2, "Deleting Objects" on page 25-10
- Section 25.5.3, "Renaming Objects and Modifying Descriptions" on page 25-10
- Section 30.15, "Enabling and Disabling Rules" on page 30-16

30.15 Enabling and Disabling Rules

For more information about this procedure, see Section 11.2.4 on page 11-2.

When you enable or disable a rule, be sure to generate logic before unit testing the configuration model. Generating logic is described in Section 28.7 on page 28-5.

You can disable and enable all rules in a Folder by selecting the Folder and then choosing the appropriate action from the **Actions** list.

To enable or disable a rule:

1. Select the rule.
2. Select **Enable** or **Disable** from the **Actions** list.
3. Click **Go**.

Alternative method:

1. Open the rule for editing.
2. Select or deselect the **Disable** check box.
3. Click **Apply**.

30.16 Reordering Rules in a Rule Sequence

When you change the order of rules in a Rule Sequence, Configurator Developer displays a message that tells you how the effective dates for each rule will be affected. You can then either make the change or cancel the operation. If you click Yes, Configurator Developer changes the effective dates of the rules. If you click No, the sequence of rules and their effective dates do not change.

For more information, see Section 18.5, "Reordering Rules and Rule Effective Dates" on page 18-3.

The procedure below assumes you are working in the Rules area of the Workbench

To change the order of rules in a Rule Sequence:

1. In the same row as the Rule Sequence you want to modify, click the icon in the **Edit** column.
2. In the same row as a rule, click the icon in the **Reorder** column.
3. In the Move Rule page, select a rule, and then click either **Move Rule Above** or **Move Rule Below**.

   Configurator Developer moves the rule to the new location in the Rule Sequence, and changes the effective dates of each rule as necessary.
4. Click **Apply**.
5. In the rule's details page, click **Apply**.

30.17 Removing Rules from a Rule Sequence

You can remove rules from a Rule Sequence at any time by selecting Move from the Actions list, or by deleting the rule. If you want to keep the rule, be sure to move it out of the Rule Sequence, rather than deleting it. Deleting a rule from a Rule Sequence also deletes the rule from the configuration model. Additionally, use caution when deleting a Rule Sequence, as this also deletes all of the rules it contains from the configuration model.

When you move a rule out of a Rule Sequence, its effective dates do not change. However, Configurator Developer adjusts the effective dates of the remaining rules in the Rule Sequence so no gaps exist between them. Additionally:

To move a rule out of a Rule Sequence:

1. In the Rules area of the Workbench, expand the Rule Sequence.
2. Select the rule in the Rule Sequence, and then select **Move** from the **Actions** list.
3. Select the destination Folder for the rule, then click **Apply**.

To remove a rule from a Rule Sequence by deleting the rule:

1. In the Rules area of the Workbench, expand the Rule Sequence.
2. Select the rule to delete, and then select **Delete** from the **Actions** list.
3. Click **Yes** to confirm the deletion.

30.18 Enabling and Disabling Rules in a Rule Sequence

When unit testing a Model, you may want to enable or disable specific rules in a Rule Sequence to view the affect at runtime. You can also disable or enable *all* rules in a Rule Sequence simultaneously, depending on your needs. Any rules that are disabled are ignored at runtime.

Before viewing the affect of your changes in a runtime UI or the Model Debugger, be sure to regenerate logic. To do this, see Section 28.7 on page 28-5.

To enable or disable only one rule in a Rule Sequence:

1. In the Rules area of the Workbench, expand the Rule Sequence and then select the rule.
2. Click the icon in the **Edit** column.

   In the rule's details page, select or deselect the **Disable** check box as appropriate.

To enable or disable *all* rules in a Rule Sequence:

1. In the Rules area of the Workbench, select the Rule Sequence.
2. Select **Disable** from the **Actions** list, and then click **Go**.

30.19 Modifying Rule Details

This section describes the settings and information that appears in a configuration rule's details page. The following sections provide a description of each setting and indicate whether it applies to only specific types of rules, or all rules.

You can define **Effectivity** and enter **Notes** for any rule. For details, see:

- Section 30.19.3, "Effectivity" on page 30-19
- Section 29.17.13, "Notes" on page 29-15

30.19.1 Definition

This section is available for all rule types and describes the selected rule by listing its participants and operator, where applicable.

How you define a rule depends on its type. For details, refer to the following sections:

- Section 30.3, "Defining Logic Rules" on page 30-3
- Section 30.4, "Defining Numeric Rules" on page 30-3
- Section 30.9, "Defining Statement Rules" on page 30-9
- Section 30.3, "Defining Logic Rules" on page 30-3
- Section 30.6, "Defining Property-based Compatibility Rules" on page 30-6
- Section 30.7, "Defining Explicit Compatibility Rules" on page 30-7
- Section 30.8, "Defining Design Charts" on page 30-8
- Section 30.10, "Creating a Rule Sequence" on page 30-9
- Section 30.11, "Creating a Configurator Extension Rule" on page 30-10

30.19.2 Violation Message

Use the settings in this section to enter text that appears when an end user makes a selection that violates the rule.

If you are viewing a rule's details page, select one of the following:

- Select **Rule Name** display only the name of the rule.
- Select **Rule Description** to display the rule's description.
- Select **Custom Text** and then enter the text you want to display at runtime when the rule is violated. Choose this option to provide specific, detailed information so the end user understands why a selection is invalid and how best to proceed.

You can enter a custom violation message for any type of rule. For an example of how a custom violation message appears in a contradiction message at runtime, see Example 11–5 on page 11-9.

Note: If you are implementing Multiple Language Support (MLS), rule violation message text is stored in a database table so it can be translated more easily. For details, see Appendix B, "Multiple Language Support".

30.19.3 Effectivity

These settings control whether a rule is active or ignored in a runtime UI, or when unit testing a configuration model using the Model Debugger. When you create a rule, its effectivity is set to Always Effective.

For general information about effectivity, see Chapter 6.

The following procedures assume you have a Model open for editing and are working in the Rules area of the Workbench.

To specify a range of effective dates for a rule:

1. Open the rule for editing.
2. In the **Effectivity** section of the rule's details page, choose **Explicit Dates** from the **Action** list, and then click **Go**.
3. Select **No Start Date** or **No End Date**, or specify a From and To date and time.

   You can specify a very wide range of dates when entering a start and end date, but this range is limited. For more information, see the *Oracle Configurator Implementation Guide*.
4. Click **Apply**.

To assign an Effectivity Set to a rule:

1. Open the rule for editing.
2. In the **Effectivity** section of the rule's details page, select **Choose Effectivity Set** from the **Action** list, then click **Go**.

   The Select Effectivity Set page lists all Effectivity Sets in the CZ schema.
3. Select an Effectivity Set, and then click **Apply**.

To assign a Usage to a rule:

1. Open the rule for editing.
2. In the **Effectivity** section of the rule's details page, click **Select Usage Exceptions**.
3. Select the Usage(s) for which the rule is not effective, and then click **Apply**.
4. In the rule's details page, click **Apply**.

For more information about Usages, see Chapter 6.

30.19.4 Unsatisfied Message

Use this section to define the message that appears when the rule is "unsatisfied" at runtime. You can specify an unsatisfied message only for Logic Rules and Comparison Rules. By default, Configurator Developer does not display a message when a rule is unsatisfied.

For more information, see Section 11.7, "Unsatisfied Rules" on page 11-15.

To create an unsatisfied rule message:

1. In the rule’s details page, specify the information to display at runtime when the rule is unsatisfied. Select **Nothing**, **Rule Name**, **Rule Description**, or **Custom Text**.

   Select **Custom Text** to enter detailed information about the rule, such as the available options that may cause it to be unsatisfied.

2. Click **Apply**.

31

User Interface Area of the Workbench

This chapter describes the tasks you can perform in the User Interface area of the Workbench.

This chapter includes the following sections:

- Creating a New User Interface
- Editing a User Interface
- Creating a User Interface Content Template
- Editing a User Interface Content Template

31.1 Introduction

You use the User Interface area of the Workbench to create and maintain all of a Model's User Interfaces. You can also unit test a UI by clicking the Test Model button at the top of the page. To navigate to the User Interface area of the Workbench, open a Model for editing and then click the User Interface link at the top of the page.

For additional information about User Interface elements and User Interface Templates, see:

- Chapter 19, "Displaying the Model"
- Chapter 20, "User Interface Templates"
- Chapter 21, "User Interface Structure and Design"

31.2 Creating a New User Interface

You use a UI Master Template to create a User Interface that is based on the Model's structure, or as the first step in building a UI from scratch.

In either case, review the predefined UI Master Templates and decide which best meets your requirements before generating a UI. See Section 20.2, "User Interface Master Templates" on page 20-2. If one of the predefined UI Master Templates can meet your requirements with only a few modifications, create a copy of the template you want to use, and then make any necessary changes. Then, select your customized UI Master Template when generating the UI.

For details, see:

- Section 31.6, "Editing a User Interface Master Template" on page 31-33
- Section 31.5, "Editing a User Interface Content Template" on page 31-2

Note: A recommended best practice is to wait until the Model's structure is complete before generating a UI. You can unit test a configuration model that does not yet have a UI using the Model Debugger. See Section 32.1, "Unit Testing Using the Model Debugger" on page 32-1.

The following procedure assumes you have a Model open for editing and are working in the User Interface area of the Workbench.

To create a new UI:

1. In the same row as the root **User Interfaces** Folder, click the icon in the **Create** column.
2. Enter a **Name** and optionally a **Description** of the UI.
3. Select a **Master Template** from the list.
4. Select one of the following **Options**:
   - Select **Generate UI for Model** to generate a UI that is based on the Model's structure.

     To show all Model structure nodes in the UI, regardless of the **Display in User Interface** setting, select **Show Entire Model**.
   - Select **Create Empty UI** if you want to build the UI from scratch.

   Note: When creating an empty UI, you cannot use a UI Master Template that provides the Dynamic Model Tree navigation style. This is because Configurator Developer generates a dynamic navigation tree according to the Model's structure, while empty UIs are not based on the Model's structure.

   ---

5. Click **Apply**.
6. If you created an empty UI, add UI content.

   For details, see Section 31.3, "Editing a User Interface" on page 31-2.

If you generated a UI that is based on the Model's structure, unit test the UI to see if any modifications are required. See Section 22.3, "Unit Testing a Generated User Interface" on page 22-3.

31.3 Editing a User Interface

Editing a UI includes adding, deleting, or modifying UI structure or specific UI elements in the User Interface area of the Workbench. These tasks may be performed when modifying a UI that is based on the Model's structure, or when you are building a UI from scratch (an empty UI). You edit all UIs in the User Interface area of the Workbench.

For a detailed example, see Section 21.17, "Designing and Creating a User Interface Page" on page 21-50.

Before modifying a User Interface, be sure the Model's structure and your configuration rules are complete.

If you created an empty UI, perform the following steps to define its content:

1. Create UI Pages and associate them with Model structure nodes.

   See Section 31.3.3, "Creating a User Interface Page" on page 31-6.

   When you create an empty UI, Configurator Developer generates a single UI Page. By default, the Associated Model Node for this Page is set to the root Model node. Configurator Developer also generates a Page Flow, which contains a Page Reference that points to the generated UI Page.

2. Add content to UI Pages.

   See Section 31.3.4, "Creating User Interface Page Content" on page 31-7.

3. Create a Menu or any required Page Flows.

   See Section 31.3.5, "Creating a Menu" on page 31-25.

4. Create Page Links (if using a Menu) or Page References (if using Page Flows) to UI Pages.

   See Section 31.3.9, "Creating a Page Reference" on page 31-27.

If you created a UI that is based on the Model's structure, refer to the following sections for common UI editing tasks:

- "Modifying the User Interface Definition" on page 31-3
- "Copying, Moving, and Deleting User Interface Elements" on page 31-5
- "Creating User Interface Page Content" on page 31-7

  This section describes how to create Layout Regions, UI Template References, and other basic elements such as Tables, Buttons, Check Boxes, Images, and so on.
- "Defining a Condition for Runtime Display and Behavior" on page 31-27
- "Sorting Options" on page 31-29
- "Assigning Actions to User Interface Elements" on page 31-30

31.3.1 Modifying the User Interface Definition

For general information about the UI Definition, see Section 21.3 on page 21-3.

The following procedure assumes you have a Model open for editing and are working in the User Interface area of the Workbench.

To modify the UI Definition:

1. In the same row as the UI Definition node, click the icon in the **Edit** column. (The name of this node is name you entered when generating the UI.)

   The UI Definition's details page appears.

2. **Primary Navigation** indicates the UI's navigation style, which is determined by the UI Master Template used to generate the UI. Examples include Step-by-Step and Model Tree.

3. The **Primary Page Flow** setting (or **Primary Menu**) determines the first page an Oracle Configurator end user sees when a configuration session begins. The name of this setting varies based on the navigation style of the UI Master Template that generated the UI. For example:

   - **Primary Page Flow** a UI whose primary navigation style is either dynamic tree or step-by-step.

- **Primary Menu** for a UI whose primary navigation style is either Single-Level or multiple-level menu.

To specify a different Page Flow or Menu, click **Choose**.

Note: You cannot change the Primary Page Flow setting if the UI was generated using a UI Master Template that provides the Dynamic Model Tree navigation style. (For example, the Oracle Browser Look and Feel with Dynamic Tree Navigation UI Master Template.)

For more information about Page Flows, see Section 21.6 on page 21-7.

4. To prevent the UI from being refreshed via the Refresh UIs button, deselect **Refresh Enabled**. See Section 28.8, "UI Refresh Status" on page 28-6.

   This setting does not prevent you from refreshing the UI from the User Interface area of the Workbench.

   For more information about this setting, see Section 19.2.4, "The Refresh Enabled Setting" on page 19-8.

5. If Configurator Developer is set up to display prices and Available to Promise (ATP) information, the **Price and Availability Display** settings determine whether this data appears in the UI, and when selling prices and ATP dates are updated.

   Setting up pricing and ATP for unit testing is described in Section 22.4, "Displaying Pricing Information and ATP Dates when Unit Testing" on page 22-3.

   Select one or more of the following settings:

   - **List Price**: Select this box to display the unit list prices for each item. This price does not change during a configuration session.
   - **Selling Price**: Select this box to display the unit selling price of all selected items. Items that are not yet selected display their unit list price. Selling prices can change during a configuration session. (See the **Recalculate Prices** setting below.)
   - **Availability**: Select this box to display ATP dates for all selected items and the entire configuration. (Note that this information can change during a configuration session.)
   - **Recalculate Prices**: If you selected the Selling Price and Availability settings (above), specify when to update prices and ATP dates at runtime:
     – Select **On Request** to update prices only when the end user chooses to do so. This is the default value. If you choose this option, you must provide a way for the user to update prices. For example, you can create a button and set its action to **Update Prices**.

     – Select **On Page Load** to recalculate prices when the end user navigates to another page, or refreshes the current page. Selecting this setting does not prevent an end user from updating prices if a UI control exists for this purpose.

     – Select **On Change** to update prices each time the end user makes a selection, enters a value, or navigates to another page.

6. The **Message Templates** settings indicate the UI Content Templates that are used to display required and optional messages at runtime.

To select a different UI Content Template, click **Choose**, select a Message Template, and then click **Apply**.

To suppress optional messages of a specific type, click **Clear** to remove the template.

The predefined Message Templates are described in Section 20.3.4 on page 20-20.

7. The **Images** settings indicate the files that are used to indicate selection state, logic selector radio buttons, and logic selector check boxes at runtime. The default values are determined by the UI Master Template you used to generate the UI.

   To use a different image at runtime, replace the existing file name with the new one, and then click **Apply**.

   Images are described in Section 20.2.2.7 on page 20-9.

8. The **Referenced User Interfaces** section lists all Model References and their associated UIs. To specify a different referenced UI, select one from the list, and then click **Apply**.

   For more information about referenced User Interfaces, see Section 4.4 on page 4-2.

9. When your changes are complete, click **Apply**.

31.3.2 Copying, Moving, and Deleting User Interface Elements

Like the Model itself, a UI appears in a hierarchical structure when viewing or editing it in the UI area of the Workbench. And, like Model structure nodes that you create in Configurator Developer, you can modify a UI by copying, moving, or deleting elements.

You copy or move UI elements by selecting them and then selecting a command from the Actions list. For details about the available actions, see Section 24.1.3, "Actions" on page 24-4. You cannot copy, move, or delete any default UI Folders (that is, the Pages, Page Flows, and Menus Folders).

When copying or moving a UI element, the node you specify as the element's new parent (its destination) must be valid for that element. Whether the new parent is a valid destination depends on its type. You can copy or move UI elements within the same UI Page or to a different Page in the same UI. You can also copy or move elements from one UI to another UI, but both UIs must belong to the same Model.

To copy or move an element:

1. Select the element.
2. Select **Copy** or **Move** from the **Actions** list, and then click **Go**.
3. If you are copying an element, select a destination for the new element.

   Accept the default value of **Current Parent** to create a copy of the selected element at its current location. Select **New Parent** to create the element in a different location. In this case, click **Choose**, select a destination, and then click **Apply**.

4. If you are moving an element, select a new parent from the list, and then click **Apply**.

To delete a UI element:

1. In the User Interface area of the Workbench, select the element to delete.
2. Select **Delete** from the Actions list, and then click **Go**.
3. Click **Yes** to confirm the action.

31.3.2.1 Converting a UI Template Reference

For general information about UI Template References, see Section 21.16.1 on page 21-48. Creating a UI Template Reference is described in Section 31.3.4.2 on page 31-7.

Use the Convert Template Reference action when you want to convert a Template Reference into a copy of the template's structure at its current location. This action changes a Template Reference into UI content that you can then modify. You may want to do this, for example, if you want to add or change something for a specific use of a template in the UI.

After converting the Template Reference to UI content, the root of the structure has the same Associated Model Node setting as the Template Reference; in other words, it is set to either Inherit from Parent or Specified (in the latter case, it is associated with a specific node). The Associated Model Node of the root element's children is set to Inherit from Parent.

The following procedure assumes you have a UI open for editing in the User Interface area of the Workbench.

To convert a Template Reference into UI content:

1. Select the Template Reference(s), then select **Convert Template Reference** from the Actions list.
2. Click **Go**, and then click **Yes** to confirm the action.

   Oracle Configurator Developer generates new content and then displays the UI structure with the selected Template Reference(s) replaced by the new inline UI content.

31.3.3 Creating a User Interface Page

For general information about UI Pages, see Section 21.4 on page 21-4.

This procedure assumes you have a UI open for editing in the User Interface area of the Workbench.

To create a UI Page:

1. In the same row as the **Pages** Folder, click the icon in the **Create** column.
2. Optionally modify the default **Name** and enter a **Description** of the Page.
3. Specify an **Associated Model Node**.

   To do this:

   a. Click **Choose Node**.
   b. Select a Model node from the list, then click **Apply**.

      For more information about associating structure nodes with UI elements, see Section 21.14 on page 21-43.

      Note: Once you save a UI Page, you cannot modify its Associated Model Node setting. Therefore, be sure to select the correct Page before saving.

      ---

4. Specify a caption for the UI Page by specifying a **Text Source**.

   For more information about UI captions, see Section 21.12 on page 21-35.

5. Optionally define a display or enabled condition for the Page.

   For details, see Section 31.3.10 on page 31-27.

6. Click **Apply**.

31.3.4 Creating User Interface Page Content

In the User Interface area of the Workbench, you can modify UI Pages by adding content such as Layout Regions, Buttons, Images, and option selection controls. For a detailed example, see Section 21.17, "Designing and Creating a User Interface Page" on page 21-50.

When creating content, the type of element you can create depends on the parent's type. In other words, it must be possible for the UI element you are creating to exist as a child of its parent. For example, almost any UI element can be a child of a Row Layout, but only a Case region can be a child of a Switcher region.

The types of UI content you can create and the kind of Model structure nodes that each type can be associated with are described in Chapter 21.

In the User Interface area of the Workbench, you can create UI content:

- Manually, by specifying the type of UI element to create, and then associating it with a Model node
- Automatically, by using a UI Content Template

The following sections describe each method.

31.3.4.1 Creating a Region from a User Interface Content Template

For general information about this task, see Section 21.16.2, "Creating UI Content from a User Interface Content Template" on page 21-50.

For information about UI Content Templates, see Section 20.3 on page 20-13.

The following procedure assumes that you have a UI open for editing in the User Interface area of the Workbench.

To create UI content using a UI Content Template:

1. In the same row as the UI element in which you want the new content to appear, click the icon in the **Create** column.
2. Select **Region from Template**, and then click **Continue**.
3. Select a UI Content Template, and then click **Apply**.

   Configurator Developer copies the content of the template into the UI structure at the location you specified.

31.3.4.2 Creating a UI Template Reference

For general information about this element, see Section 21.16.1, "User Interface Template References" on page 21-48.

To convert an existing Template Reference into UI content, see Section 31.3.2.1 on page 31-6.

To create a Template Reference:

1. In the same row as a Page Region element, click the icon in the **Create** column.
2. Select **Other**, and then select **UI Template Reference**.

3. Click **Continue**.
4. Select the UI Content Template you want to use, and then click **Apply**.

   A UI Template Reference appears in the UI structure at the location you specified.
5. Optionally modify the following settings:
   - Associated Model Node (see Section 21.15 on page 21-44)
   - Refresh Enabled (see Section 19.2.4 on page 19-8)
   - Conditional Presentation (see Section 21.11 on page 21-33)
6. Click **Apply**.

31.3.4.3 Modifying a UI Template Reference

Modify a UI Template Reference when you want it to refer to a different UI Content Template.

For additional information, see Section 21.16.1, "User Interface Template References" on page 21-48.

To modify a UI Template Reference:

1. In the same row as a UI Template Reference, click the icon in the **Edit** column.
2. In the Template Reference’s details page, click **Choose Template**.

   The template name is a link that you can click to view or edit the template. When you do this, the template opens for editing in the User Interface area of the Workbench.
3. Select the UI Content Template you want to refer to, and then click **Apply**.
4. In the template reference’s details page, click **Apply**.

31.3.4.4 Creating a Layout Region

For general information about Layout Regions, including a description of each type, see Section 21.8, "Layout Regions" on page 21-8.

The type of Layout Region you can create depends on its parent.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a Layout Region:

1. In the same row as the UI element in which you want the Layout Region to appear, click the icon in the **Create** column.
2. Select **Basic Layout Region**, **Node List Layout Region**, or **Message List Layout Region**.
3. Select a Layout Region type from the list. For example, Stack Layout, Bulleted List, Table Layout, Row Layout, Flow Layout, Cell Format, or Header Region.
4. Click **Continue**.

   The Layout Region’s details page appears. This page contains different settings depending on the type of element you are creating.
5. Optionally modify the default **Name**.
6. If you are creating a Message List Layout Region, select a Message Type.
7. Define the following settings, where applicable:

- Associated Model Node (see Section 21.14 on page 21-43)
- Caption Source (see Section 21.12 on page 21-35)
- Formatting (see Section 31.3.12 on page 31-28)
- Sorting (see Section 31.3.13 on page 31-29)
- Conditional Presentation (see Section 31.3.10 on page 31-27)
- Refresh Enabled (see Section 19.2.4 on page 19-8)

8. Click **Apply**.

31.3.4.5 Creating an Item Selection Table

For general information about this element, see Section 21.10.5 on page 21-30.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

You can also generate this element automatically using one of the predefined BOM Item Table Control templates. For details, see Section 31.3.4.1 on page 31-7.

> **Tip:** If you want to create this element manually, review the structure of one of the predefined BOM Item Table Control templates in Configurator Developer before you begin. These templates are described in Section 20.3.3.1 on page 20-17.

To create an Item Selection Table:

1. In the same row as the layout UI element in which you want the table to appear, click the icon in the **Create** column.
2. In the Create UI Content page, select **Other**, and then select **Item Selection Table** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   You can associate an Item Selection Table only with a BOM Model, BOM Option Class or Option Feature.
6. Select a **Selection Style** to specify the type of selection control that appears in the first column of the table at runtime.

   For example, if the node associated with this element contains mutually exclusive options and you want the control to indicate selection state, select **Single-select, Enhanced Radio Button**. For details, see:

   - Section 21.9.11, "Check Box" on page 21-22
   - Section 21.9.12, "Enhanced Check Box" on page 21-23
   - Section 21.9.14, "Radio Button" on page 21-24
   - Section 21.9.15, "Enhanced Radio Button" on page 21-24
7. In the **Formatting** section, optionally modify the width of the table.

   For details, see Section 31.3.12 on page 31-28.
8. Optionally specify a **Sorting** method for the table rows.

   For details, see Section 31.3.13 on page 31-29.

9. Optionally define a **Display Condition**, an **Enabled Condition**, or both for the table itself and the table rows.

   For details, see Section 21.11 on page 21-33.

10. Click **Apply**.

11. Create the table columns. This step is required before you can unit test the UI.

    For example, to display columns with the headings "Description" and "Configure", create the following elements as children of the Item Selection Table element, and associate them with the same node as the table itself:

    - A Styled Text element: For the **Text Source**, select **Associated Model Node System Property**, and then select **Description**. Enter "Description" in the **Table Column Header** field.

      See Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14.

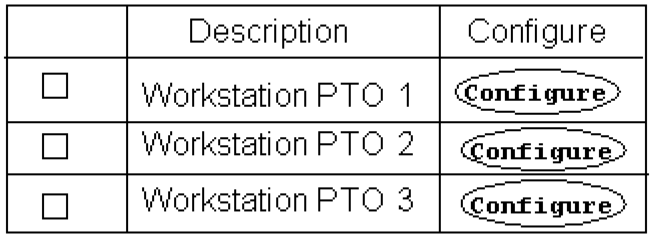
    - A Button element (to enable the end user to navigate to and configure configurable items that appear in the table): For the **Text Source**, select **Text Expression** and then enter "Configure" in the field provided. Then, enter "Configure Item" in the **Table Column Header Text** field. Finally, define a **Button Action** of **Configure Associated Subcomponent**.

      See Section 31.3.4.18, "Creating a Custom Button" on page 31-18.

    Figure 31–1 shows how this table might appear at runtime.

    At runtime, Oracle Configurator generates columns for each element you created, and displays a separate row for each of the associated Model node's children.

Figure 31–1 Item Selection Table



	Description	Configure
☐	Workstation PTO 1	Configure
☐	Workstation PTO 2	Configure
☐	Workstation PTO 3	Configure

See also Section 31.3.4.6, "Creating an Instance Management Table" on page 31-10.

> **Tip:** If you do not want to display a button for all of the items in the table, create a Flow Layout element beneath the Item Selection Table first, and then create the Button element as a child of the Flow Layout.
>
> When you create a Button directly beneath the table element, the column containing the Button appears only if the first row in the table contains a Button. In the example above, the Configure column appears only because the row contains a Configure Button (since the first item in the table is configurable).

31.3.4.6 Creating an Instance Management Table

For general information about this element, see Section 21.10.6 on page 21-30.

The procedure below describes how to create an Instance Management Table manually. To create this element using a UI Content Template, see Section 31.3.4.1 on page 31-7.

> **Tip:** If you want to create this element manually, review the structure of the predefined Instance Management Table template in Configurator Developer before you begin. This template is described in Section 20.3.3.3 on page 20-17.

The following procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create an Instance Management Table:

1. In the same row as the Layout Region in which you want the Instance Management Table to appear, click the icon in the **Create** column.
2. In the Create UI Content page, select **Other**, then select **Instance Management Table** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   You can associate an Instance Management Table only with an instantiable Component or instantiable Model Reference node.
6. To display a button in the table that enables an end user to add instances of the associated Model node at runtime, select **Show Add Instances Button**, and then define the button's UI caption.

   For details about defining a UI caption, see Section 21.12 on page 21-35.
7. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.
8. Optionally specify a **Sorting** method for the table rows.

   For details, see Section 31.3.13 on page 31-29.
9. Optionally define a **Display Condition**, an **Enabled Condition**, or both for the table itself, and the table rows.

   For details, see Section 21.11 on page 21-33.
10. Click **Apply**.
11. Create the table columns. For examples, see Section 31.3.4.5 on page 31-9.

    This step is required before you can unit test the UI.

    > **Tip:** If you want end users to be able to modify the name of each instance at runtime, create a Text Input element as the first child beneath the Instance Management Table element. For the Text Input element's **Display System Property** setting, select **InstanceName**.

31.3.4.7 Creating a Connection Navigator Table

For general information about this element, see Section 21.10.7 on page 21-31.

The procedure below describes how to create a Connection Navigator Table manually. You can also create this element automatically using a UI Content Template. For details, see Section 31.3.4.1 on page 31-7.

> **Tip:** If you want to create this element manually, review the structure of the predefined Connection Navigator template in Configurator Developer before you begin. See Section 20.3.3.13, "Connection Navigator Template" on page 20-20.

The following procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Connection Navigator Table:

1. In the same row as the layout element in which you want the Connection Navigator Table to appear, click the icon in the **Create** column.
2. In the Create UI Content page, select **Other**, then select **Connection Navigator Table** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.
6. Optionally specify **Sorting** options for the rows in the table (that is, each connected component).

   For details, see Section 31.3.13 on page 31-29.
7. Optionally define a **Display Condition**, an **Enabled Condition**, or both for the table itself or the rows in the table.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.
9. Create a Styled Text element as a child of the Connection Navigator Table element. At runtime, this text is the table's column header. See Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14.

   When creating this element, specify the following:

   - **Associated Model Node Property**: Display Name
   - **Link Action Type**: Navigation (Configure Associated Subcomponent)

   At runtime, the names of each connected component appear as links the user can click to navigate to the connected component.
10. If you want to display additional content in the table, create elements under the Connection Navigator Table element. At runtime, each additional element appears in a separate column.

    For example, to indicate each connected component's selection state, create a Selection Status Indicator element under the Styled Text element. For details, see Section 31.3.4.28, "Creating a Status Indicator or Unsatisfied Status Indicator" on page 31-22.

31.3.4.8 Creating a Summary Table

For general information about this element, see Section 21.10.4 on page 21-29.

The procedure below describes how to create a Summary Table manually. You can also create this element automatically using a UI Content Template. For details, see Section 31.3.4.1 on page 31-7.

Tip: If you want to create this element manually, review the structure of one of the predefined Summary Table templates in Configurator Developer before you begin. For details, see Section 20.3.5.2, "Summary Page Templates" on page 20-22.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Summary Table element:

1. In the same row as a layout element, click the icon in the **Create** column.
2. Select **Other**, and then select **Summary Table** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. In the **Hierarchy Column Contents** section, specify a **Text Source**, **Style**, and **Table Column Header** for the hierarchy column.

   For details about these settings, see Section 21.12 on page 21-35.
6. Specify display and formatting settings for the table. These settings include:
   - **Width (percent)**: The width of the table as a percentage of the width of its container. Leave this field blank if you want the size of the table to be determined by its contents.
   - **Expand all levels on entry**: Select this setting to display all selected options and their parent nodes in the table when the end user navigates to the UI page. In other words, the hierarchy of selected options is expanded by default.
   - **Dynamically update Summary data**: Select this setting if you want the Summary Table to be updated automatically when the end user updates the configuration.

     If you do not select this setting, Oracle Configurator updates the table only when the page is redisplayed; for example, when an end user navigates to the UI Page.

     For example, you may want to select this setting to create a region that displays a "running summary" in a page that also contains controls for selecting options. Do not select this setting if you are creating a Summary page or template that does not provide option selection controls.
7. Optionally define a **Display Condition** for the table itself, or for rows within the table.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.
9. Create the table columns.

   This step is required before you can unit test the UI.

   For examples, see Section 31.3.4.5 on page 31-9.

31.3.4.9 Creating a Cell Format

Create a Cell Format to create and format cells within a Row Layout region. Row Layouts are described in Section 21.8.4 on page 21-12. For general information about the Cell Format element, see Section 21.8.6 on page 21-14.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Cell Format:

1. In the same row as a Row Layout element, click the icon in the **Create** column.
2. Select **Basic Layout Region**, and then select **Cell Format** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.
7. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 31.3.10 on page 31-27.
8. Click **Apply**.

31.3.4.10 Creating a Styled Text Element

For general information about this element, see Section 21.9.1 on page 21-17.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Styled Text element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Styled Text** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Optionally modify the default **Text Source** setting and specify a **Text Style**.

   For details, see Section 21.12 on page 21-35.
7. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.

31.3.4.11 Creating a Static Styled Text Element

For general information about this element, see Section 21.9.2 on page 21-18.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Static Styled Text element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select Basic Element, and then select **Static Styled Text** from the list.

3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Enter a **Prompt** and then optionally specify a **Text Style**.

   For details, see Section 21.12 on page 21-35.
7. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.

31.3.4.12 Creating a Formatted Text Element

For general information about this element, see Section 21.9.3 on page 21-18.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Formatted Text element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Other**, and then select **Formatted Text** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally modify the default **Text Source** setting and specify a **Text Style**.

   For details, see Section 21.12 on page 21-35.

   If you enter a text expression, note that Configurator Developer does not validate the syntax of the text you enter. Therefore, be sure to enter well-formed, valid HTML.
7. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.

31.3.4.13 Creating a Text Link Element

For general information about this element, see Section 21.9.4 on page 21-19.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Text Link element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Other**, and then select **Text Link** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

6. Optionally modify the default **Text Source** and **Rollover Source** settings and specify a **Text Style**.

   For details, see Section 21.12 on page 21-35.

   If you enter a text expression, note that Configurator Developer does not validate the text's syntax. Therefore, be sure to enter well-formed, valid HTML.

7. Define a **Link Action**.

   For details, see Section 31.3.15 on page 31-30.

8. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.

9. Click **Apply**.

31.3.4.14 Creating a Raw Text Element

For general information about this element, see Section 21.10.9 on page 21-32.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Raw Text element:

1. In the same row as any UI element, click the icon in the **Create** column.

2. Select **Other**, and then select **Raw Text** from the list.

3. Click **Continue**.

4. Optionally modify the default **Name**.

5. Optionally modify the **Associated Model Node** setting.

6. Optionally modify the default **Text Source** setting, specify a **Text Style**, and define a **Link Action**.

   For details, see Section 21.12 on page 21-35.

   If you enter a text expression, note that Configurator Developer does not validate the syntax. Therefore, be sure to enter well-formed, valid HTML.

7. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.

8. Click **Apply**.

31.3.4.15 Creating an Image Element

For general information about this element, see Section 21.9.5 on page 21-19.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create an Image element:

1. In the same row as any UI element, click the icon in the **Create** column.

2. Select **Basic Element**, and then select **Image** from the list.

3. Click **Continue**.

4. Optionally modify the default **Name**.

5. Optionally modify the **Associated Model Node** setting.

For details, see Section 31.3.14 on page 31-30.

6. Optionally modify the default **Image Source** and **Rollover Text Source** settings.

   For details, see Section 21.12 on page 21-35.

7. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.

8. Click **Apply**.

31.3.4.16 Creating an Image Button

For general information about this element, see Section 21.9.6 on page 21-20.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create an Image Button:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Image Button** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.

6. Optionally modify the default **Image Source** and **Rollover Text Source** settings.

   For details, see Section 21.12 on page 21-35.

7. Optionally define a **Link Action**.

   For details, see Section 31.3.15 on page 31-30.

8. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.

9. Click **Apply**.

31.3.4.17 Creating a Standard Button

For general information about this element, see Section 21.9.7 on page 21-20.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Standard Button:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Standard Button** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.

6. Specify the Button’s action, UI caption, and access keys by selecting a **Button Type**.

For a description of these actions and the access keys for each, see Table 21–1, " Common Runtime Actions and Access Keys" on page 21-37.

A Standard Button's UI caption is the same as its type. For example, if the Button Type is Apply, the Button's UI caption is "Apply".

7. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.

8. Click **Apply**.

31.3.4.18 Creating a Custom Button

For general information about this element, see Section 21.9.8 on page 21-21.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Custom Button:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Custom Button** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.

6. Optionally modify the default **Text Source** and **Rollover Text Source** settings.

   For details, see Section 21.12 on page 21-35.

7. Optionally define a **Button Action**.

   For details, see Section 31.3.15 on page 31-30.

8. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.

9. Click **Apply**.

31.3.4.19 Creating a Spacer

For general information about this element, see Section 21.9.9 on page 21-21.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Spacer element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Other**, and then select **Spacer** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.

6. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.

7. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.

8. Click **Apply**.

31.3.4.20 Creating a Separator

For general information about this element, see Section 21.9.10 on page 21-22.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Separator:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Separator** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.

6. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.

7. Click **Apply**.

31.3.4.21 Creating a Check Box

For general information about this element, see Section 21.9.11 on page 21-22.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Check Box element:

1. In the same row as a UI Page or layout element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Check Box** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.

7. Click **Apply**.
8. Optionally create another element to display a UI caption with the Check Box. For example, create either a Styled Text or Static Styled Text element. For details, see:

   - Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14
   - Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14

31.3.4.22 Creating an Enhanced Check Box

For general information about this element, see Section 21.9.12 on page 21-23.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create an Enhanced Check Box element:

1. In the same row as a layout UI element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Enhanced Check Box** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
7. Click **Apply**.
8. Optionally create a Spacer element and another element to display a UI caption with the Enhanced Check Box. For example, create a Styled Text or Static Styled Text element. See:
   - Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14
   - Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14

31.3.4.23 Creating an Instantiation Check Box

For general information about this element, see Section 21.9.13 on page 21-23.

To create an Instantiation Check Box:

1. In the same row as a layout UI element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Instantiation Check Box** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
7. Click **Apply**.
8. Optionally create another element to display a UI caption with the Instantiation Check Box. For example, create a Styled Text or Static Styled Text element. See:
   - Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14
   - Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14

31.3.4.24 Creating a Radio Button

For general information about this element, see Section 21.9.14 on page 21-24.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Radio Button:

1. In the same row as a layout UI element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Radio Button** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
7. Click **Apply**.
8. Optionally create another element to display a UI caption for the Radio Button. For example, create a Styled Text or Static Styled Text element. See:
   - Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14
   - Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14

31.3.4.25 Creating an Enhanced Radio Button

For general information about this element, see Section 21.9.15 on page 21-24.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create an Enhanced Radio Button:

1. In the same row as a layout UI element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Enhanced Radio Button** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
7. Click **Apply**.
8. Optionally create another element to display a UI caption for the Enhanced Radio Button. For example, create a Styled Text or Static Styled Text element. See:
   - Section 31.3.4.10, "Creating a Styled Text Element" on page 31-14
   - Section 31.3.4.11, "Creating a Static Styled Text Element" on page 31-14

31.3.4.26 Creating a Drop-down List

For general information about this element, see Section 21.9.16 on page 21-24.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Drop-down List element:

1. In the same row as a layout UI element (for example a Stack Layout or Row Layout), click the icon in the **Create** column.
2. Select **Basic Element**, and then select **Drop-down List** from the list.

3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally modify the default **Text Source** setting.

   For details, see Section 21.12 on page 21-35.
7. Optionally enter an Excluded Item Prefix, an Excluded Item Prefix, or both.

   Use these settings to indicate which items that appear in a drop-down list are excluded from the configuration at runtime. For example, if you enter "X -" in the Excluded Item Prefix field, each excluded item in the drop-down list appears as follows at runtime:

   ```
   X - Item Display Name
   ```

   Oracle Configurator inserts a space between the item's display name and the prefix or suffix at runtime.
8. Optionally specify a **Sorting** method for options that will appear in the list.

   For details, see Section 31.3.13 on page 31-29.
9. Optionally define a **Display Condition**, an **Enabled Condition**, or both for the Drop-down List element itself. You can also click **List Item Display Condition** to define a display condition for items in the list.

   For details, see Section 21.11 on page 21-33.
10. Click **Apply**.

31.3.4.27 Creating a Text Input Element

For general information about this element, see Section 21.9.17 on page 21-25.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Text Input element:

1. In the same row as a valid parent UI element, click the icon in the **Create** column.

   For a list of valid parent UI elements, see Section 21.9.17 on page 21-25.
2. Select **Basic Element**, and then select **Text Input** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.
6. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.
7. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.

31.3.4.28 Creating a Status Indicator or Unsatisfied Status Indicator

For general information about these elements, see Section 21.9.18 on page 21-26.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Selection Status Indicator or Unsatisfied Indicator element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Element**, and then select either **Selection Status Indicator** or **Unsatisfied Indicator** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Optionally define a **Display Condition**.

   For details, see Section 21.11 on page 21-33.
7. Click **Apply**.

31.3.4.29 Creating a Content Container

For general information about this element, see Section 21.10.3 on page 21-29.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Content Container element:

1. In the same row as any UI element, click the icon in the **Create** column.
2. Select **Basic Layout Region**, and then select **Content Container** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Optionally modify the default **Formatting** settings.

   For details, see Section 31.3.12 on page 31-28.
7. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
8. Click **Apply**.

31.3.4.30 Creating Switcher and Case Regions

For general information about Switcher and Case Regions, see Section 21.10.2 on page 21-27.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a Switcher Region:

1. In the same row as a UI Page, click the icon in the **Create** column.
2. Select **Other**, then select **Switcher Region** from the list.
3. Click **Continue**.

4. Optionally modify the default **Name**.
5. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
6. Optionally modify the **Object** setting.

   Select one of the following:
   - Select **Bound** to use the Switcher element's Associated Model Node.
   - Select **Other Model Node** if you want to associate the Switcher with a different structure node.
   - Select **Session Data** to associate the Switcher with the status of the configuration.
7. Select a **Property** of the object specified in the previous step.

   For details, see Section 21.10.2, "Switcher and Case Regions" on page 21-27.
8. Click **Apply**.
9. Create a Case region (see below).

To create a Case region:

1. In the same row as a Switcher region, click the icon in the **Create** column.

   The Case region's details page appears. The default **Switcher Object** and **Switcher Property** are derived from the Case region's parent (that is, the Switcher region).
2. Optionally modify the **Associated Model Node** setting.

   For details, see Section 31.3.14 on page 31-30.
3. Optionally modify the default **Case Value** by selecting a new value from the list, or by entering a value.

   If the Switcher Property is **Selection State**, the **Value** setting is a drop-down list. If the Switcher Property is **Value** (that is, a Property value), then the **Value** setting is an input field.
4. Click **Apply**.

31.3.4.31 Creating a HideShow Region

For general information about HideShow Regions, see Section 21.8.8 on page 21-16.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a HideShow Region:

1. In the same row as a UI Page, click the icon in the **Create** column.
2. Select **Basic Layout Region**, and then select **HideShow** from the list.
3. Click **Continue**.
4. Optionally modify the default **Name**.
5. Optionally modify the default **Associated Model Node**.

   For details, see Section 21.14 on page 21-43.
6. Specify a Text Source for the element's expanded and collapsed states.

   For details, see Section 21.12 on page 21-35.

7. If you want the region to be collapsed the first time an end user views the page in which it appears, deselect the **Initially Expanded** check box.
8. Optionally enter the style sheet definition you want to use to render the caption text in the **Style** field.

   By default, the style sheet BLAF.xss determines, the font, size, and color of the text. This style sheet is located in the `oa_html/cabo/styles` directory.
9. Optionally define a **Display Condition**, an **Enabled Condition**, or both.

   For details, see Section 21.11 on page 21-33.
10. Click **Apply**.

31.3.5 Creating a Menu

For general information about Menus, see Section 21.5 on page 21-5.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a Menu:

1. In the same row as the **Menus** Folder, click the icon in the **Create** column.
2. Enter a **Name** and optionally a **Description**.
3. Select a Menu **Type**:
   - Single Level Side Menu (see Section 20.2.7 on page 20-12)
   - Multi-level Side Menu (see Section 20.2.8 on page 20-12)
   - Model Tree Side Menu (see Section 20.2.6 on page 20-11)
   - Subtab Layout (see Section 20.2.9 on page 20-12)
4. To prevent the Menu from being updated when you refresh the UI, deselect **Refresh Enabled**.

   For details, see Section 19.2, "Refreshing a User Interface" on page 19-2.
5. Enter any additional information in the **Notes** field.
6. Click **Apply**.
7. Create the Menu’s Page Links.

   For details, see Section 31.3.7, "Creating a Page Link" on page 31-26.

31.3.6 Creating a Menu Label

Create a Menu Label to group a set of Page Links within a Multi-Level Menu. For general information about Menus, see Section 21.5.1 on page 21-7.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a Menu Label:

1. In the same row as a Multi-Level Menu, click the icon in the **Create** column.
2. Select **Label**, and then click **Next**.
3. Enter a **Name**.
4. Optionally modify the **Associated Model Node** setting.

For details, see Section 21.14 on page 21-43.

5. Define the **Label Caption**.

   For details about each setting, see Section 21.12 on page 21-35.

6. Click **Finish**, or click **Finish and Create Another** to create another Menu Label.

   The Label you created appears in the UI area of the Workbench as a child of the Menu in which it will appear at runtime.

   To group Page Links, create them as children of the Menu Label. See Section 31.3.7, "Creating a Page Link" on page 31-26.

31.3.7 Creating a Page Link

For general information about Menus and Page Links, see Section 21.5 on page 21-5.

You can group Page Links within a Multi-Level Menu by creating each group as children of a different Label Menu. See Section 31.3.6, "Creating a Menu Label" on page 31-25.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Page Link:

1. In the same row as a Menu or Menu Label, click the icon in the **Create** column.
2. If you are creating the Page Link within a Menu and the Menu's type is Multi-Level, select **Page Link**, and then click **Continue**.
3. Select the UI Page that will be the Page Link's target, and then click **Next**.
4. Enter a **Name**.
5. In the **Contents** section, specify how the runtime Oracle Configurator creates the Page Link's UI caption (that is, the link text).

   For details about each setting, see Section 21.12 on page 21-35.

6. Click **Apply**, or **Apply and Create Another**.

   The Page Link you created appears in the UI' structure as a child of the Menu (or Menu Label) in which it will appear at runtime.

31.3.8 Creating a Page Flow

For general information about Page Flows, see Section 21.6 on page 21-7.

This procedure assumes you have a UI open for editing in the UI area of the Workbench.

To create a Page Flow:

1. In the same row as the **Page Flows** Folder, click the icon in the **Create** column.
2. Optionally modify the default **Name**.
3. To prevent the Page Flow from being updated when you refresh the UI, deselect **Refresh Enabled**.

   For details, see Section 19.2.4, "The Refresh Enabled Setting" on page 19-8.

4. Enter any additional information in the **Notes** field.
5. Click **Apply**.

6. Create Page References.

   For details, see Section 31.3.9, "Creating a Page Reference" on page 31-27.

31.3.9 Creating a Page Reference

Create Page References to incorporate UI Pages into a Page Flow. For general information about Page References and Page Flows, see Section 21.6 on page 21-7.

This procedure assumes you are modifying a UI in the User Interface area of the Workbench.

To create a Page Reference:

1. In the same row as a Page Flow, click the icon in the **Create** column.
2. Enter a **Name**, and then select a target UI Page.
3. Click **Apply**, or **Apply and Create Another**.

   The Page Reference appears in the UI area of the Workbench as a child of the Page Flow.

31.3.10 Defining a Condition for Runtime Display and Behavior

For an overview of both display and enabled conditions, see Section 21.11 on page 21-33.

The following procedure assumes you are working in the User Interface area of the Workbench.

To define a display or enabled condition for a UI element:

1. In the same row as the element for which you want to define a runtime condition, click the icon in the **Edit** column.

   The element's details page appears.
2. In the **Conditional Presentation** region, click **Define** next to **Display Condition** or **Enabled Condition**.

   If the selected UI element is a Drop-down List, you can optionally define a display condition for individual items in the list. To do this, click **Define** next to **List Item Display Condition**.
3. In the Define Condition page, select an **Object**. This determines whether the condition refers to a Property of:

   - The UI element's **Bound Model Node** (that is, its associated node)
   - A node that you specify (**Other Model Node**)
   - Configuration **Session Data**

   If you are defining a List Item Display Condition, the default Object is **List Item Model Node** and you cannot change it.
4. Specify which **Property** of the Object you want Oracle Configurator to use when evaluating the condition. The options available in this list depend on the selected Object.

   If the Object is a Model node, choose one of the node's System Properties from the list. If the node can have a logic state at runtime, you can also choose **Selection State**, **Logic State**, or **State**.

If the node is a Numeric Feature, it cannot have a logic state. In this case, select **Value**, and then specify a value in the **Value** field (see below).

If the Object is **Session Data**, select a configuration session property. For example, **Valid**, **Unsatisfied**, or **ModelQuantity**. For a complete list and a description of each, see Section 5.4, "Configuration Session Properties" on page 5-8.

5. Indicate how the selected Property is evaluated against the value you specified by selecting a **Comparison** of **Is** or **Is Not**.
6. Select or enter a **Value**.

   For example, if Property is set to **Logic State**, select **Selected**, **Selectable**, or **Excluded**. If Property is set to **Value**, enter a value.
7. Click **Apply**.
8. In the element's detail page, click **Apply** to save the condition.

31.3.11 Refreshing a User Interface

For a general information about refreshing a UI, see Section 19.2 on page 19-2.

The following procedure assumes you have a Model open for editing and are working in the User Interface area of the Workbench.

To refresh a User Interface:

1. In the same row as the UI you want to refresh, click the icon in the **Refresh** column.
2. Click **Yes**.

To refresh all User Interfaces that have changed since they were created or last refreshed:

1. Open the Model for editing.
2. In the General area of the Workbench, click **Refresh UIs**. See Section 28.8, "UI Refresh Status" on page 28-6.

31.3.12 Formatting User Interface Elements

A UI element's details page contains various formatting settings that determine the element's appearance at runtime. The prompt for each setting indicates the unit of measure used at runtime. For example, **Display Length (in lines)**.

The following procedure assumes you are working in the User Interface area of the Workbench.

To format a UI element:

1. In the same row as the element you want to modify, click the icon in the **Edit** column.

   The element's details page appears.
2. Modify the **Formatting** settings.

   Depending on the selected element's type, the following settings may be available:

   - **Display Length**: How much horizontal space to allocate for this element.
   - **Display Height**: How much vertical space to allocate for this element.

- **Width**: The selected element's width, as a percentage, relative the UI page or table. This value is relative to either the UI page or a table, depending on the selected element's type.
- **Horizontal Alignment** and **Vertical Alignment**: The location of the selected element in a UI page or within a table. Choose **Left**, **Center**, **Right**, **Start**, or **End**.
- **Number of rows to display at a time**: The maximum number of rows to display in a table. When a table contains more than the number of rows specified here, a separate UI control is provided at runtime to view the additional rows.
- **Password-style**: This setting is available only for a Text Input UI element. Select this box if you want Oracle Configurator to display any text that the end user enters as asterisks (for example, "******"). You may want to do this if the element is used to collect sensitive information, such as an end user's credit card number.

3. Click **Apply** to save your changes.

31.3.13 Sorting Options

When editing some elements, such as an Item Selection Table or a Drop-down List, you can specify the order in which the element's options appear at runtime. In an Item Selection Table, the Sorting settings determine the order in which the table rows appear. In a Drop-down List, these settings determine the order in which the options appear in the list.

If you define a sorting method for a UI element, the runtime Oracle Configurator updates the order in which the element's options appear when the end user refreshes the current page, or navigates to another page. For example, if the sorting method is based on a the value of the `Quantity` System Property and the quantity of an option in an Item Selection List changes, Oracle Configurator reorders the options when the end user navigates to a different page.

The following procedure assumes you are working in the User Interface of the Workbench.

To modify the order in which rows appear in a table or options appear in a list:

1. In the same row as the element, click the icon in the **Edit** column.

   The element's details page appears.

2. Modify the **Sorting** settings.

   These settings include:

   - **Sort Field**: Select **No Sorting** to list options in the same order as they appear in the Model structure.

     Select **Item Caption** to sort options alphabetically using their display name (see **Sort Order**, below).

     Select **List Item System Property** or **List Item User Property** to sort options alphabetically using one of the associated node's Properties (see **Sort Order**, below). These settings are labeled Row Item System Property and Row Item User Property when creating or editing a table element.

   - **Sort Order**: This setting controls whether Oracle Configurator sorts options in ascending or descending order, based on the **Sort Field** setting.

3. Click **Apply** to save your changes.

31.3.14 Changing the Associated Model Node

For general information about the relationship between Model structure nodes and UI elements, see Section 21.14 on page 21-43.

The following procedure assumes you are working in the User Interface area of the Workbench.

To associate a Model node with a UI element, or modify an existing association:

1. In the same row as the element, click the icon in the **Edit** column.

   The element's details page appears.

2. Select one of the following settings for the **Associated Model Node**:

   - **Inherited from Parent**: This associates the UI element with its parent Model node. This is the default setting.
   - **Specified**: Select this setting and then click **Choose Node** to associate the UI element with a Model node that you specify.
   - **None**: Select this setting if you do not want to associate the UI element with a node or any session data. If you choose this setting, be sure the element's caption and rollover text sources are set to use a Configurator Session Property or a Text Expression. See Section 21.12 on page 21-35.

   The list of available settings depends on the selected UI element's type.

3. Click **Apply** to save your changes.

31.3.15 Assigning Actions to User Interface Elements

For general information about actions, see Section 21.13 on page 21-38.

The following procedure assumes you are working in the User Interface area of the Workbench.

To assign an action to a UI element:

1. In the same row as the element, click the icon in the **Edit** column.

   The element's details page appears.

2. Next to the *Element Type* **Action** setting, click **Define**.
3. Select an **Action Type** and an associated action.

   For a description of each action, see Section 21.13 on page 21-38.

4. If you selected an action of **Start Page Flow**, click **Choose Page Flow** and then select a Page Flow from the list.
5. If you selected an **Action Type** of either Open URL or **Raise Command Event**, see Section 21.13.1, "Action Parameters" on page 21-41 for details about the additional parameters that are required.
6. Click **Apply** to save your changes.

31.4 Creating a User Interface Content Template

For an overview of UI Content Templates, see Section 20.3 on page 20-13.

You can create a UI Content Template either by copying one of the predefined UI Content Templates or create it from scratch. In each case, you begin the process in the Main area of the Repository.

You may want to create a UI Content Template from scratch if your business requires reusable UI content that cannot be created simply by modifying one of the predefined templates. When you create a template from scratch, you must then open it for editing in the User Interface area of the Workbench and define its content (for example, by creating some of the UI elements described in Chapter 21, "User Interface Structure and Design").

If you created a template by copying one of the predefined templates, you may want to modify some of the template's existing elements, add new elements, or modify the template's structure. For details, see Section 31.5, "Editing a User Interface Content Template" on page 31-33.

> **Tip:** Before you begin, review the structure and content of a predefined template that is of the same type as the template you want to create. For example, if you want to create a template to display instantiable components in the UI, create a copy of the predefined Instance Management Control Template, and then open it for editing in the User Interface area of the Workbench. You may also want to generate a test UI to review how the predefined template displays content at runtime.

For an example of how custom UI Content Templates can be used in a UI, see Section 21.17, "Designing and Creating a User Interface Page" on page 21-50.

31.4.1 Creating a User Interface Content Template

For general information about UI Content Templates, see Section 20.3 on page 20-13.

To create a UI Content Template by copying an existing template:

1. In the Main area of the Repository, select the UI Content Template you want to copy.
2. Select **Copy** from the **Actions** list.
3. Specify the destination for the new template, then click **Finish**.
4. Open the new template for editing, and modify it as required.

   See Section 31.5, "Editing a User Interface Content Template" on page 31-33.

Alternative Method - Creating a UI Content Template from Scratch

1. In the Main area of the Repository, in the same row as the Folder in which you want to store the new template, click the icon in the **Create** column.
2. Select **UI Content Template**, and then click **Continue**.

   Refer to Section 31.4.2, "Common Steps for Creating a User Interface Content Template" on page 31-31 for the remaining steps.

31.4.2 Common Steps for Creating a User Interface Content Template

The steps below assume you are creating a UI Content Template from scratch and are viewing the template's details page.

To complete the template's definition:

1. Select a **Template Type**.

   Configurator Developer uses this setting to determine which templates appear when you are creating or modifying a UI Master Template. For example, you are modifying a UI Master Template and want to select a different Utility Template for the Two-Page Flow Navigation setting. When you click the **Choose** button, Configurator Developer displays only templates whose type is Utility Template.

2. If you selected a **Template Type** of **Control Template**, **Message Template**, or **Utility Template**, select a specific Control, Message, or Utility template type from the list.
3. Click **Continue**.
4. Enter a **Name**, **Description**, and any **Notes**.

   A UI template's name must be unique within the Folder in which you are creating it.

5. Make a selection for each remaining setting.

   For details about each setting, see Section 31.4.3, "User Interface Content Template Settings" on page 31-32.

6. Click **Apply**.
7. From the Main area of the Repository, open the template for editing. The structure of the template appears in the User Interface area of the Workbench.
8. Define the template's content.

   For example, if you created a Button Bar template, define the buttons, their labels, and a runtime action for each. For details, see Section 31.3.4, "Creating User Interface Page Content" on page 31-7.

9. When the template is complete, click **Apply** to save it.

31.4.3 User Interface Content Template Settings

This section describes the various settings that are available when you are creating or editing a UI Content Template.

The settings that appear depend on the template's type and may include the following:

- **Root Layout Region**: Select **Basic Layout Region** or **Node List Layout Region**. For details about this setting, see Section 21.8 on page 21-8.
- **Layout Style**: This setting enables Configurator Developer to group and display templates with compatible Layout Styles together. For example, when creating UI content using a template, Configurator Developer displays all available templates in a table. Templates with compatible Layout Styles appear sequentially in the table. For example, select **Label/Data Row Layout** or **Multi-column Row Layout**. Select **Other** if you do not want to group this template.
- **Title Text**: The text that appears in the message header at runtime. This setting is available for some Message Templates.
- **Presentation**: This setting determines whether the message appears as a Modal Message Box or a Dialog Page at runtime. This setting is available for some Message Templates.

  For a description of each message type, see Section 20.3.4 on page 20-20.

- **Button Bar Template**: If the **Presentation** setting is **Dialog Page**, click **Choose** to select a Button Bar Template.

For details about this type of template, see Section 20.3.2 on page 20-15.

31.5 Editing a User Interface Content Template

The predefined UI Content Templates are read-only. The procedure below assumes you are editing a template that you created either from scratch or by copying one of the predefined templates.

Editing a UI Content Template is similar to editing a UI. For example, you open the template for editing in the User Interface area of the Workbench, and then create, move, and modify UI elements that comprise the template.

By default, you can edit a UI Content Template only if it is not locked or it is locked by you. For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.

To modify a UI Content Template:

1. From the Main area of the Repository, expand the Folder in which the template is stored.
2. In the same row as the template, click the icon in the **Locking** column.

   For details, see Section 24.2, "Locking Models and UI Content Templates" on page 24-5.
3. In the same row as the template, click the icon in the **Edit** column.

   The template opens for editing in the User Interface area of the Workbench.
4. Modify the template by adding, moving, or modifying UI elements.

   For details, see Section 31.3.4, "Creating User Interface Page Content" on page 31-7.

31.6 Editing a User Interface Master Template

The predefined UI Master Templates are read-only. The procedure below assumes you have either created a UI Master Template from scratch or by copying one of the predefined templates. See Section 25.3.7, "Creating a User Interface Master Template" on page 25-5.

To edit a UI Master Template:

1. From the Main area of the Repository, expand the Folder in which the template is stored.
2. In the same row as the template, click the icon in the **Edit** column.
3. Modify the template by navigating to each section and changing the pagination and layout settings, specifying different UI Content Templates, and so on.

   For more information, see Section 20.2.2, "UI Master Template Information and Settings" on page 20-4.

32 Model Debugger and User Interface Testing

This chapter describes how to unit test a configuration model using either the Model Debugger or a generated User Interface.

This chapter includes the following sections:

- Unit Testing Using the Model Debugger
- Unit Testing Using a Generated User Interface

For general information about unit testing, see Chapter 22, "Testing and Debugging".

32.1 Unit Testing Using the Model Debugger

For general information about the Model Debugger, see Section 22.2 on page 22-2.

32.1.1 Launching the Model Debugger from Configurator Developer

To launch the Model Debugger from Configurator Developer:

1. Open a Model for editing, and then navigate to the Structure, Rules, or UI area of the Workbench.
2. Click **Test Model**.
3. Choose whether to **Create a new configuration** or **Restore a saved configuration**, and then click **Next**.
4. If you chose to restore a saved configuration, select a saved configuration from the list, and then click **Next**.

   You can sort the data in each column by clicking the column header.
5. Verify that **Model Debugger** is selected, and then optionally enter any **Session Parameters**.

   For details, see Section 22.1.1.1, "Session Parameters" on page 22-2.
6. Click **Finish**. The Model Debugger displays the selected Model and applies the session parameters that you specified.

   For details on using the Model Debugger, see Section 32.1.3 on page 32-2.

32.1.2 Launching the Model Debugger from the E-Business Suite Home Page

You can launch the Model Debugger directly from the Oracle E-Business Suite Home page, without starting Configurator Developer. The only limitation when launching the Model Debugger this way is that you can restore a saved configuration, but you cannot create a new configuration.

To launch the Model Debugger from the Oracle Applications E-Business Suite Home page:

1. Select the **Oracle Configurator Developer** responsibility, and then select **Test Configuration**.
2. Enter a Configuration **Header ID** and **Revision Number**, and then click **Next**.
3. If you want to display pricing and ATP information in the Model Debugger, click **Preferences**, and then enter the callback interface packages and procedures to use in the **Custom Initialization Parameters** field. For details, see Section 22.4 on page 22-3.
4. To load the saved configuration using the latest, published version of the Model:
    a. Select **Use Currently Published Model Version**.

       The Model may have been modified since the configuration was saved. Select this option to test the saved configuration against the most up-to-date version of the Model structure and rules.

    b. Specify the publication you want to test by specifying a date, Application Code, Publication Mode, or Usage.
    c. Click **Next**.
5. To test the version of the Model against which the configuration was saved, choose **Use Model Version Saved in Configuration**, and then click **Next**.

   Select this option to load the Model in the same state it was in when the configuration was saved. In other words, the Model structure and rules that are loaded are the same as when the configuration was saved, regardless of whether the published version has changed.
6. Optionally enter any **Session Parameters**.

   For details, see Section 22.1.1.1, "Session Parameters" on page 22-2.
7. Click **Finish**.

   For details on using the Model Debugger, see Section 32.1.3 on page 32-2.

32.1.3 Using the Model Debugger

Following are some suggestions for unit testing a configuration model using the Model Debugger.

- Click **Show Legend** to see the icons that the Model Debugger uses to indicate each option's logic state and any options that are unsatisfied. Expand parts of the Model structure to view their children, or click **Expand All** to view the Model's entire structure. Click **Collapse All** to view only the first level of the Model structure.
- Click the icon in the **Add to Watch List** column to monitor the status of specific items while unit testing. All items added to the Watch List appear in a separate table at the top of the page, along with their selection state and current value. Each item in the Watch List still appears in the Model structure. You can add items to the Watch List or remove them at any time while using the Model Debugger. To remove a node from the Watch List, click the icon in the **Remove** column.
- Select and enter values for various items to add them to the configuration. Also add component instances and configure them. Notice how these actions affect the configuration and items in your Watch List.

- To run a Configurator Extension, click the icon in the **Run Extension** column. If multiple Configurator Extensions exist for the selected node, choose the one to run from the page that appears, and then click **Apply**.
  - The number and distribution of Configurator Extension icons is affected by the choice of Model Node and Event Binding Scope in the Configurator Extension Rules defined for the Model. For example, if the Event Scope was chosen as Global, then Configurator Extension icons appear for every node of the Model. If the Event Scope was chosen as Base Node, then Configurator Extension icon appears only on that node of the Model.
  - Only Configurator Extensions bound to the `onCommand` event can be explicitly run in the Model Debugger. When you run one, the Model Debugger prints a message displaying the command string that was handled and the names of the Configurator Extension Rules that handled it. If multiple commands are bound to the same node, then the Model Debugger presents a list of them so that you can select the one that you want to run.
- Navigate to the **Summary** tab to view a summary of all selected items. If pricing is enabled, this page also displays item pricing and Available to Promise (ATP) information.

  For details about pricing and ATP in the Model Debugger, see Section 22.4 on page 22-3.
- Navigate to the **Status** tab to view items based on their current status in the configuration.

  For example, go to the **Unsatisfied Items** section to see a list of all items that contain required selections, or go to **User Requests** to view only items that you selected during the debugging session (in other words, not items selected by the propagation of a rule). When testing a saved configuration, the **User Requests** section also shows items selected when the debugging session started.
- For details about using the **View** list, see Section 24.1.1 on page 24-2.
- Click **Save** to save the configuration periodically. This performs an intermediate save, but does not give you the option to exit the Model Debugger.
- Click **Revert to Saved** to rollback the changes and return the configuration to the way it was before the last time you saved it.
- To exit the Model Debugger without saving the configuration, click **Cancel**.
- To save the configuration and exit the Model Debugger:
  1. Click **Finish**.
  2. Enter a **Configuration Name**.
  3. To create a new configuration with a unique Configuration Header ID, select **New Configuration** from the **Save As** list. (You can save a new configuration only if you launched the Model Debugger from Configurator Developer.)

     To increment the Revision Number and save the configuration with the same Configuration Header ID, select **New Revision**.
  4. Click **Apply**.
  5. If the configuration is invalid or incomplete, the Model Debugger displays a status message. Click **Yes** to continue.
  6. Make a note of the **Configuration Header ID** and **Revision Number** for future reference.

7. If you launched the Model Debugger from Configurator Developer, click **Return to Configurator Developer**.

   If you launched the Model Debugger from the E-Business Suite Home page, click **Home** to return to the E-Business Suite Home page, or **Logout** to exit Oracle Applications.

32.2 Unit Testing Using a Generated User Interface

To generate a User Interface, see Section 31.2, "Creating a New User Interface" on page 31-1.

32.2.1 Launching a Runtime User Interface from Configurator Developer

To unit test a Model in a generated User Interface:

1. Open the Model for editing, then navigate to the Structure, Rules, or UI area of the Workbench.
2. Click **Test Model**.
3. Choose either **Create a new configuration**, or **Restore a saved configuration**, then click **Next**.
4. If you chose to restore a saved configuration, select it from the list, and then click **Next**.

   If you chose to create a new configuration, go to the next step.
5. Select **Model UI**, and then select a UI from the list.
6. Specify **Session Parameters**, and then click **Finish**. Session parameters are described in Section 22.1.1.1 on page 22-2.
7. Unit test the Model structure, rules, and review the UI to ensure that it performs as expected. For more information see Section A.3, "Configuring an Item in a Runtime Oracle Configurator" on page A-3.

32.2.2 Launching a Runtime User Interface from the E-Business Suite Home Page

To launch a saved configuration from the Oracle Applications E-Business Suite Home page, you must know the configuration's header ID and Revision number.

When you open a saved configuration from the E-Business Suite Home page, you can view only the UI that was saved with the configuration; in other words, you do not have the option to select one of the Model's UIs.

To unit test a saved configuration from the E-Business Suite Home page:

1. Click **Test Configuration.**
2. Enter the saved configuration's **Header ID** and **Revision Number**.
3. To test the published version of the Model, choose **Use Currently Published Model Version**.

   To test the version of the Model against which the configuration was saved, choose **Use Model Version Saved in Configuration**, and then click **Next**.
4. If you chose to test the currently published Model version, enter applicability parameters to specify the publication you want to test.
5. Specify **Session Parameters**, and then click **Finish**.

Session parameters are described in Section 22.1.1.1 on page 22-2.

For details about unit testing a configuration in a User Interface, see Section A.3, "Configuring an Item in a Runtime Oracle Configurator" on page A-3.

Part VII

Appendices

Part VII contains the following appendices:

A
The Runtime Oracle Configurator

This appendix describes the runtime Oracle Configurator, and contains the following sections:

A.1 Overview

An Oracle Configurator allows you to configure a product by selecting from a list of available options. Oracle Configurator ensures that the configuration is valid and orderable by enforcing constraints that govern how all selected items fit together. Oracle Configurator therefore reduces configuration errors, which in turn reduces change order processing and the amount of rework required in manufacturing.

With Oracle Configurator you can:

- Validate configurations
- Automatically select configuration options

Oracle Configurator supports:

- Assemble-to-Order (ATO) and Pick-to-Order (PTO) BOM Models
- User-defined item attributes
- Flexible configuration constraints
- Integrated configuration validation
- Automatic configuration completion
- Enterprise Resource Planning (ERP) integration
- Customer Relationship Management (CRM) integration

To determine whether a host application supports an Oracle Configurator, see the *About Oracle Configurator* documentation for this release on Metalink, Oracle's technical support Web site.

A.2 The Oracle Configurator Window

The Oracle Configurator window may display the item you are configuring in a generated HTML UI, the Generic Configurator User Interface, a Java applet UI, or a DHTML UI. The type of User Interface that appears depends on whether the item was published after being imported into Oracle Configurator Developer and, if it was, the version of Configurator Developer in which it was generated.

The layout and method of configuring a product vary depending on which type of user interface is presented.

In a Generic Configurator UI or Java applet:

- Only Bills of Material items are available for selection.
- Expand configurable items to view, select, and enter quantities from optional items.
- In the Java applet, a summary of all items included in the configuration appears in a separate frame. In the Generic Configurator UI, click **Preview Configuration** at any time to view a summary of your selections.

For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

In a DHTML or generated HTML UI:

- Some needs-assessment questions may appear to gather information about how you will use the product. Your answers typically default some options and exclude others from the configuration.
- Navigate to the next component to be configured using the navigation controls provided on each page.
- Configure each component on a separate page.
- Click UI controls provided to view a summary of all items included in the configuration, and to return to the configuration.

Implementors set values for specific profile options to determine how Oracle Applications access the Oracle Configurator window. For information about the profile options that affect Oracle Configurator, see the *Oracle Configurator Installation Guide*.

A.2.1 Keyboard Access in the Oracle Configurator Window

Oracle Configurator enables end users with disabilities to navigate the Configurator window using only the keyboard. For example, end users can navigate to each option in the Configurator window by pressing the Tab key. Typically, keyboard UI navigation travels from left to right and from the top to the bottom of the page, and each option is highlighted when it can be selected.

Table A–1 on page A-2 lists the available keystrokes and the corresponding actions in the Configurator window. These commands perform the same action in both the Generic Configurator UI and User Interfaces generated in Configurator Developer.

Table A–1 Keyboard Access in the Configurator Window

Press this ...	If you want to ...
Tab	Shift the focus forward to each option in the UI (from left to right, top to bottom of the page). Shift the focus to the next frame. For example, in a DHTML UI, from the navigation tree to the selection area of the page.

Table A–1 (Cont.) Keyboard Access in the Configurator Window

Press this ...	If you want to ...
Shift+Tab	Shift the focus to each option in the reverse order through the UI (right to left, bottom to top).
	Navigate to the previous frame.
Enter	Select an option (that is, add it to the configuration) or execute an action (for example, activate the **Finish** button to commit the configuration).
	Expand or collapse the selected subtree.
Space Bar	Toggle the state of a check box. For example, change the status from selected (true) to deselected (false).
Up and Down Arrow Keys	Shift the focus through each option in a list.
	Shift the focus up or down links in the navigation tree.
Left and Right Arrow Keys	Scroll to the left or right.
Delete	Deselect a selected option.

A.3 Configuring an Item in a Runtime Oracle Configurator

Oracle Configurator validates each selection against the rules defined for the item that you are configuring. Your selections can trigger configuration rules that automatically exclude or make available other options in the product. You may see this occur on the current page when you select an option.

You configure each component on a separate Oracle Configurator page. Selectable options may be presented in the form of needs assessment questions that you answer to provide basic requirements for the product.

The UI provides controls that enable you to switch from Configuration mode to the Configuration Summary page. This page displays information about all items selected during the configuration session, such as each item's name, description, quantity selected, and pricing information (if enabled). For more information, see Section 19.5, "The Configuration Summary Page" on page 19-11.

If you make a selection that violates a configuration rule, a message describes the violation and provides suggestions on how to proceed.

To configure an item:

1. Begin selecting from the list of available options, or answer any needs-assessment questions that appear.

   For more information, see Section A–2, " Selecting Options in a Runtime Oracle Configurator" on page A-4.

2. As you make selections, they are automatically validated against the rules that have been defined for the Model or item that you are configuring.

   If you change one of your selections, it can automatically change the choice of valid selections for other features of the product, according to the configuration rules. You see those changes when you select an affected feature.

   If you make a selection that violates a configuration rule, Oracle Configurator displays a message describing the violation and your options for proceeding.

3. When you have finished configuring a component, navigate to another component and continue making selections.

For more information about navigation during a configuration session, see Table A–1, " Keyboard Access in the Configurator Window" on page A-2.

4. If controls are provided to do so, optionally view the Available To Promise (ATP) dates and pricing information for specific items.
5. Optionally navigate to the Configuration Summary page to review a summary of the configuration.
6. When you are finished, save the configuration using the provided UI control. For example, click the Done, Finish, or Save button.

Warning: Canceling a configuration session instead of using the provided UI controls to save or end the session causes all selections to be lost. For example, Oracle Configurator does not save the configuration if you click the Cancel button or close your Web browser by choosing File > Close.

7. If the configuration is valid and complete, a message presents options for proceeding.

   If you launched Oracle Configurator from a host application such as Oracle Order Management and want to save the configuration, acknowledge the message to send the configuration data to the host application for processing. At this point, the host application closes the Oracle Configurator window.

   If you are unit testing a generated User Interface and want to save the configuration, use the control provided to return to Oracle Configurator Developer.

   If you do not want to save the configuration, use the control provided to discard your changes and return to the application from which you launched Oracle Configurator.
8. If the configuration is incomplete or invalid, a message indicates that the configuration requires additional selections. In this case, you can do one of the following:
   - Return to the configuration to select additional options
   - Ignore the message and save the configuration in its current state. You can restore the configuration later and select any required options.

Table A–2 Selecting Options in a Runtime Oracle Configurator

To select this ...	Do this ...
One option	Select the check box for each desired option.
One option in a group of mutually exclusive options	Select the check box for the desired option. (This list might be presented as a set of radio buttons.)
One or more options in a group that allows multiple selections	Select the check boxes for all desired options.
A quantity for a numeric option	Enter the desired quantity. Entering a quantity greater than zero automatically selects the option.

A.4 Configuring an Order from a Bill of Materials

In Oracle Order Management, you can configure products created from a BOM Model in Oracle Bills of Material when:

- You have not installed Oracle Configurator.

  In this case, you select options from an ATO or PTO BOM Model using the Order Management Options window.

- Oracle Configurator is installed, but no publication exists for the selected ATO or PTO BOM Model.

  In this case, the Generic Configurator UI appears and you make selections from the BOM as it was defined in Oracle Bills of Material.

- There is no Oracle Configurator servlet running.

  In this case, you configure the item by selecting options from the Order Management Options window.

- The relevant Oracle Applications profile options are set appropriately. For more information, see the *Oracle Configurator Installation Guide*.

To more information, see Section A.3, "Configuring an Item in a Runtime Oracle Configurator" on page A-3.

A.5 Preconfiguring an Item

You can create a configured bill of material for a pre-defined ATO item by invoking Oracle Configurator directly from Oracle Bills of Material. You may want to do this, for example, when the exact same configuration of an item is ordered frequently.

For more information about preconfiguring items, see the *Oracle Bills of Material User's Guide*.

A.6 Creating Instances at Runtime

For background information, see Chapter 7, "Instantiation".

A button or other UI control may be provided to allow you to add one or more instances of a specific component to the configuration. The UI caption may appear as a link that you can click to navigate to the UI Page for that instance, or you may be able to enter a new name. For details about how default instance names are generated, see Section 28.9, "Runtime Display Names" on page 28-6.

If a component is selected by default when the configuration session begins, it is required in the configuration. If a quantity input field appears with a component, the default value (if any) is the Default Quantity that is defined in Oracle Bills of Material.

Each time an end user adds an instance, the runtime Oracle Configurator appends a number to the newly instantiated component's name. For example, if the component's caption is "Port", the new instances are labeled "Port [1]", "Port [2]", and so on. Note that if the end user later deletes a component, it is possible that the instance numbers will no longer appear sequentially. See Figure 20–4, "An Instance Management Table at Runtime" on page 20-18.

If the instance being configured is not complete and the end user adds another instance, then Oracle Configurator creates the new instance. In other words, an instance does not have to be complete before the end user can create another. However, an end user can create new instances only if the maximum number of instances allowed in the configuration has not yet been reached. For details, see Section 7.4, "Modifying Instantiability" on page 7-3.

When you save a configuration that is valid but has incomplete components, Oracle Configurator displays a message listing the incomplete items. However, you may choose to continue and pass the saved configuration back to Oracle Order Management. When you restore a saved configuration in which some instances were left in an incomplete state, those instances are preserved in the restored configuration.

Because configured items with incomplete components cannot be manufactured, the batch validation process displays errors when an Oracle Order Management user tries to book the order. For more information about batch validation and booking orders, see the *Oracle Order Management User's Guide*.

Note: You cannot add component instances in the Generic Configurator UI. For details about the Generic Configurator UI, see the *Oracle Configurator Implementation Guide*.

A.6.1 Behavior of Instances

You can change the quantity of a specific component in Oracle Configurator and on the line item for that component in Oracle Order Management. The number of component instances that exist in a configuration can only be modified in Oracle Configurator.

Pricing and Available To Promise (ATP) information are done per instance. For more information about pricing and ATP, see the *Oracle Configurator Implementation Guide*.

The Quantity Cascade calculations for BOM items is preserved within any configuration of an instance. For more information about Quantity Cascade, see Section 11.3, "Imported BOM Rules" on page 11-3.

B

Multiple Language Support

This appendix describes Multiple Language Support (MLS) and things to consider when implementing MLS with Configurator Developer and a runtime Oracle Configurator.

This appendix includes the following sections:

- BOM Item Descriptions
- Translatable Model Text
- User Interface Captions
- Runtime Messages
- Languages Setting
- Unit Testing a Translated User Interface
- Publishing and Multiple Language Support

B.1 Introduction

If you implement Multiple Language Support (MLS), you can create a Model and one or more User Interfaces in your base language, and then display the runtime UI in any language in which you do business.

Following is an overview of the process:

1. In Oracle Inventory, enter alternate translations for all Item descriptions.

   See Section B.2, "BOM Item Descriptions" on page B-2.

2. In Configurator Developer, set Runtime Display Names to Description for both BOM and non-BOM nodes.

   See Section B.4, "User Interface Captions" on page B-2.

3. Extract all text from the CZ_LOCALIZED_TEXTS table, translate it into all required languages, and then re-upload the translated text to the database.

   For details, see the *Oracle Configurator Implementation Guide*.

4. Unit test your User Interface in a different language.

   See Section B.7, "Unit Testing a Translated User Interface" on page B-4.

B.2 BOM Item Descriptions

When defining an Item in Oracle Inventory, a user can enter an alternate translation of the Item's description in any installed language.

Each Oracle Applications installation has one base language, but can have additional installed languages. Oracle Inventory users can enter alternate translations for each Item description using that application, but the item *name* exists only in the base language of the Oracle Applications installation. (Oracle Inventory users cannot enter translated descriptions for Item names.)

For example, the base language of your Oracle Applications installation is English, but French, Spanish, and German are installed. When creating a new Item, an Oracle Inventory user enters the Item name and description in English, then uses the Translation window to enter alternate descriptions in French, Spanish, and German. For more information, see the *Oracle Applications User's Guide*.

When you import a BOM Model, all alternate translations for Item descriptions are also imported into the CZ schema. A Model-specific setting enables you to use this text to create UI captions when generating a User Interface in Configurator Developer. For details, see Section B.4, "User Interface Captions" on page B-2.

B.3 Translatable Model Text

Any text that you enter in Configurator Developer that can appear in a runtime UI is stored in the CZ_LOCALIZED_TEXTS database table. To deploy a UI in multiple languages, you must extract the text from this table, translate it into one or more other languages, and then upload the alternate translations to the database. (This is described in detail in the *Oracle Configurator Implementation Guide*.) Depending on which language is specified at runtime, the translated text is used when unit testing a UI from Configurator Developer, or by a host application in a runtime Oracle Configurator.

Text copied into CZ_LOCALIZED_TEXTS includes:

- BOM and non-BOM Model node descriptions.

  Model node *names* are not stored in this table, and therefore are not translatable.

  See Section B.2, "BOM Item Descriptions" on page B-2.
- Rule violation and other runtime messages.

  See Section B.5, "Runtime Messages" on page B-3.
- Unsatisfied rule messages.

  See Section 30.19.4, "Unsatisfied Message" on page 30-19.
- User Interface captions.

  See Section B.4, "User Interface Captions" on page B-2.
- Translatable Text Property values.

  See Section 5.6, "Property Data Types" on page 5-11.

B.4 User Interface Captions

The Runtime Display Names setting in the General area of the Workbench controls how Configurator Developer generates default UI captions for all Model structure nodes. To ensure that UI captions can be translated and displayed in other languages at runtime, be sure to set this option to Description for both BOM and non-BOM nodes. For details, see Section 28.9, "Runtime Display Names" on page 28-6.

You can specify a different source for a UI element's caption when editing a UI. However, an element's caption is stored in CZ_LOCALIZED_TEXTS and can be translated only if the text is derived from one of the following sources:

- The `DisplayName` System Property (this is the default)

  See Section 5.6, "Property Data Types" on page 5-11.

- A User Property whose data type is Translatable Text

  See Section 5.6, "Property Data Types" on page 5-11.

- A custom text expression

  See Section B.4.1, "Text Expressions" on page B-3.

B.4.1 Text Expressions

When you enter a custom text expression for a UI element's caption, Configurator Developer copies the text to CZ_LOCALIZED_TEXTS. Therefore, the text that you enter is translatable.

Text expressions can also contain System or User Properties, but not all Properties are translatable. For example, you enter the following text expression:

If you `select &Description, additional options may be available for &Name.`

In this example, `&Description` and `&Name` are System Properties, but only `&Description` is translatable (because all node descriptions are copied to CZ_LOCALIZED_TEXTS).

When using Properties in a text expression, remember that only the following are stored in CZ_LOCALIZED_TEXTS and are therefore translatable:

- `&DisplayName`
- `&Description`
- Any User Property whose data type is Translatable Text (for example, `&Color`)

For more information, see Section 21.12.1, "Defining a Text Expression" on page 21-36.

B.5 Runtime Messages

The runtime Oracle Configurator uses the `DisplayName` System Property when creating violation and other messages. For example, if you choose to derive `DisplayName` from the node name, the violation message at runtime displays the name of the Feature, BOM Option Class, or Option node that caused the violation.

Oracle Configurator also uses this System Property to create the text of violation messages. You specify what text is used to create a violation message when defining rules and Resources. For details, see Section 30.19.2 on page 30-18.

You must manually translate any messages that Configurator Extensions display at runtime. For more information about Configurator Extensions, see Chapter 17, "Configurator Extensions".

B.6 Languages Setting

The Languages setting in the Preferences page controls the language in which all prompts, instructional text, and so on appear in the Configurator Developer user interface. This setting is user-specific, and displays a list of all languages that are installed at your site. To view the Preferences page, click the Preferences global button. For more information, see Section 24.3.3 on page 24-8.

The Languages setting also controls which language is displayed when unit testing a generated User Interface from Configurator Developer (that is, after all required text has been translated). To view a Model in a different language at runtime, change the Languages setting in the Preferences page before launching the User Interface. See Section 32.2, "Unit Testing Using a Generated User Interface" on page 32-4.

B.7 Unit Testing a Translated User Interface

If you have implemented MLS and want to use the same UI with multiple languages, be sure to thoroughly review the layout of each UI page when unit testing and then make changes as necessary in the User Interface area of the Workbench. Some changes may be needed because the organization and content of the UI may vary depending on the specified language.

For example:

- The length of text descriptions can vary significantly between languages
- Some languages are read from right to left, rather than left to right (for example, Arabic)

Note: Some UI elements have Start and End settings. The runtime UI uses these settings to horizontally align text based on the specified language, and to reverse the display of text if the specified language is read from right to left. For details, see Section 31.3, "Editing a User Interface" on page 31-2.

B.8 Publishing and Multiple Language Support

For general information about publishing, see Chapter 23.

When creating a publication request, you use the Languages applicability parameter to control in which languages the publication can be displayed at runtime. For details, see Section 23.5, "Applicability Parameters" on page 23-4. Running the Oracle Applications concurrent program to create the publication also copies all translated text to the target database.

When a host application that is part of the Oracle E-Business Suite launches an Oracle Configurator, the host application's Language setting indicates the end user's preferred language. If a publication is found for the specified language, and all other applicability parameters also match, the publication is displayed in the requested language. Otherwise, the runtime Oracle Configurator displays an error.

For example, the Language for an Oracle Order Management user's session is set to French. When the user clicks the Configure button in the Sales Orders window, a request to view the publication in French is created and passed to the database. The publication's Languages applicability parameter includes French, so when the configuration model and UI appears in the Oracle Configurator window, all text appears in the end user's preferred language.

After publishing a Model, you can modify the list of languages in which a publication is available by modifying the Language applicability parameter. For example, a publication's Language applicability parameter includes German, French, and English. You edit the publication in Configurator Developer, deselect French, and then click Apply. As a result, the publication is no longer available when an Oracle Order Management user requests the publication in French (in this case, the host application displays an error). Editing a publication is explained in Section 27.6 on page 27-6.

For more information about the database tables used when publishing, see the *Oracle Configurator Implementation Guide*.

C Rules, Node Types, and System Properties

This appendix lists which Model structure nodes and System Properties are valid when defining a Logic, Numeric, or Comparison Rule. For general information about System Properties, see Section 5.3, "System Properties" on page 5-2.

C.1 Node Types and Valid System Properties when Defining Rules

When defining a rule, you specify which Model structure nodes participate in the rule. When defining a Logic Rule, Numeric Rule, or Comparison Rule, you can also specify whether a node's System Property participates in the rule.

For example, you can contribute a value to a participating node's runtime quantity by defining a Numeric Rule and selecting the node's `Quantity` System Property. For more information, see Section 13.7, "Using Properties when Defining a Numeric Rule" on page 13-4.

This section present which nodes and System Properties can participate as:

- Logic Rule: First Operand and Second Operand
- Numeric Rule: First Operand
- Numeric Rule: Second Operand
- Comparison Rule: First Operand
- Comparison Rule: Second Operand

Table C–1 summarizes all available node types and System Properties that you can use when defining Logic, Numeric, or Comparison Rules. When reading the table, consider the following:

- When using *any_node*.Selection(), the node must be mutually exclusive.
- When using *any_node*.State(), the node in First Operand is converted to a numeric value.
- Using *any_node*.State() in Second Operand causes an error when generating logic.
- A blank cell in the table indicates that the node or System Property is not valid, and is therefore not available when defining the rule.

Table C–1 Rules, Valid Node Types, and System Properties

Node Type	**Logic**	**Numeric**	**Numeric**	**Comparison**	**Comparison**
	First Operand & Second Operand	**First Operand**	**Second Operand**	**First Operand**	**Second Operand**
any_node.Property(*true/false*)	Yes	(converted to numeric)		(converted to numeric)	Yes
any_node.Property(*numeric*)		Yes		Yes	
Required or Optional BOM Model	Yes	Yes	Yes	Yes	Yes
Required or Optional BOM Model.Options()	Yes	Yes			Yes
Required or Optional BOM Model.Selection()	Yes	Yes		Yes	Yes
Required or Optional BOM Model.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Required or Optional BOM Model.Quantity()		Yes	Yes	Yes	
Required or Optional BOM Model.MinInstances()		Yes	Yes (optional BOM Model only)	Yes	
Required or Optional BOM Model.MaxInstances()		Yes		Yes	
Required or Optional BOM Model.InstanceCount()		Yes		Yes	
Multiple BOM Model	Yes	Yes	Yes	Yes	Yes
Multiple BOM Model.Options()	Yes	Yes			Yes
Multiple BOM Model.Selection()	Yes	Yes		Yes	Yes
Multiple BOM Model.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Multiple BOM Model.Quantity()		Yes	Yes	Yes	
Multiple BOM Model.MinInstances()		Yes	Yes	Yes	
Multiple BOM Model.MaxInstances()		Yes	Yes	Yes	
Multiple BOM Model.InstanceCount()		Yes		Yes	
Option Class	Yes	Yes			Yes
Option Class.Options()	Yes	Yes			Yes
Option Class.Selection()	Yes	Yes			Yes
Option Class.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Option Class.Quantity()		Yes	Yes	Yes	
Standard Item	Yes	Yes	Yes	Yes	Yes
Standard Item.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Standard Item.Quantity()		Yes	Yes	Yes	

Table C–1 (Cont.) Rules, Valid Node Types, and System Properties

Node Type	**Logic**	**Numeric**	**Numeric**	**Comparison**	**Comparison**
	First Operand & Second Operand	**First Operand**	**Second Operand**	**First Operand**	**Second Operand**
Optional NonBOM Model.MinInstances()		Yes	Yes	Yes	Yes
Optional NonBOM Model.MaxInstances()		Yes		Yes	Yes
Optional or Multiple NonBOM Model.InstanceCount()		Yes		Yes	
Multiple NonBOM Model.MinInstances()		Yes	Yes	Yes	
Multiple NonBOM Model.MaxInstances()		Yes	Yes	Yes	
Optional Component.MinInstances()		Yes	Yes	Yes	Yes
Optional Component.MaxInstances()		Yes		Yes	Yes
Optional or Multiple Component Model.InstanceCount()		Yes		Yes	
Multiple Component.MinInstances()		Yes	Yes	Yes	
Multiple Component.MaxInstances()		Yes	Yes	Yes	
Boolean or Option Feature	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Boolean Feature.State() or Count Feature.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Count Feature	Yes	Yes	Yes	Yes	Yes
Count Feature.Quantity()		Yes	Yes	Yes	
Option Feature.Options()	Yes	Yes			Yes
Option Feature.Selection()	Yes	Yes		Yes	Yes
Option Feature.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Option	Yes	(converted to numeric)	Yes	(converted to numeric)	Yes
Counted Option	Yes	Yes	Yes	Yes	Yes
Option.State() or Counted Option.State()	Yes	(converted to numeric)	(logic generation error)	(converted to numeric)	Yes
Option.Quantity() or Counted Option.Quantity()		Yes	Yes	Yes	
Integer or Decimal Feature		Yes	Yes	Yes	
Integer or Decimal Feature.Value()		Yes	Yes	Yes	
Total or Resource		Yes	Yes	Yes	
Total.Value() or Resource.Value()		Yes	Yes	Yes	
Constant		Yes		Yes	

C.1.1 Logic Rule: First Operand and Second Operand

The following nodes and System Properties are valid participants on both sides of a Logic Rule:

any_node.Property(*true/false*)
Required, Optional, or Multiple BOM Model
Required, Optional, or Multiple BOM Model.Options()
Required, Optional, or Multiple BOM Model.Selection() [Note: Mutually exclusive only.]
Required, Optional, or Multiple BOM Model.State()
Option Class
Option Class.Options()
Option Class.Selection()
Option Class.State()
Standard Item
Standard Item.State()
Boolean or Option Feature
Boolean Feature.State() or Count Feature.State()
Count Feature
Option Feature.Options()
Option Feature.Selection()
Option Feature.State()
Option
Counted Option
Option.State() or Counted Option.State()

The value of operands in a Logic Rule are Boolean or a Boolean Collection.

C.1.2 Numeric Rule: First Operand

The following nodes and System Properties are valid participants on the First Operand side of a Numeric Rule:

any_node.Property(*true/false*) [Note: Converted to a numeric value.]
any_node.Property(*numeric*)
Required, Optional, or Multiple BOM Model
Required, Optional, or Multiple BOM Model.Options()
Required, Optional, or Multiple BOM Model.Selection() [Note: Mutually exclusive only.]
Required, Optional, or Multiple BOM Model.State() [Note: Converted to a numeric value.]
Required or Optional BOM Model.Quantity()
Required or Optional BOM Model.MinInstances()
Required or Optional BOM Model.MaxInstances()
Required or Optional BOM Model.InstanceCount()
Option Class
Option Class.Selection()
Option Class.State() [Note: Converted to a numeric value.]
Option Class.Quantity()
Standard Item
Standard Item.State() [Note: Converted to a numeric value.]
Standard Item.Quantity()
Optional NonBOM Model.MinInstances()
Optional NonBOM Model.MaxInstances()
Optional or Multiple NonBOM Model.InstanceCount()
Multiple NonBOM Model.MinInstances()

Multiple NonBOM Model.MaxInstances()
Optional Component.MinInstances()
Optional Component.MaxInstances()
Optional or Multiple Component Model.InstanceCount()
Multiple Component.MinInstances()
Multiple Component.MaxInstances()
Boolean or Option Feature [Note: Converted to a numeric value.]
Boolean Feature.State() or Count Feature.State() [Note: Converted to a numeric value.]
Count Feature
Count Feature.Quantity()
Option Feature.Selection()
Option Feature.State() [Note: Converted to a numeric value.]
Option [Note: Converted to a numeric value.]
Counted Option
Option.State() or Counted Option.State() Note: Converted to a numeric value.]
Option.Quantity()or Counted Option.Quantity()
Integer or Decimal Feature
Integer or Decimal Feature.Value()
Total or Resource
Constant
Total.Value() or Resource.Value()

The value of First Operand in a Numeric Rule is Decimal.

C.1.3 Numeric Rule: Second Operand

The following nodes and System Properties are valid participants on the Second Operand side of a Numeric Rule:

Required, Optional, or Multiple BOM Model
Required, Optional, or Multiple BOM Model.State() [Note: Causes an error when generating logic.]
Required, Optional, or Multiple BOM Model.Quantity()
Multiple BOM Model.MinInstances()
Multiple BOM Model.MaxInstances()
Option Class
Option Class.State() [Note: Causes an error when generating logic.]
Option Class.Quantity()
Standard Item
Standard Item.State() [Note: Causes an error when generating logic.]
Standard Item.Quantity()
Multiple NonBOM Model.MinInstances()
Multiple NonBOM Model.MaxInstances()
Multiple Component.MinInstances()
Multiple Component.MaxInstances()
Boolean or Option Feature [Note: Causes an error when generating logic.]
Boolean Feature.State() or Count Feature.State() [Note: Causes an error when generating logic.]
Count Feature
Count Feature.Quantity()
Option Feature.Options()
Option Feature.Selection()
Option Feature.State() [Note: Causes an error when generating logic.]
Option
Counted Option

Option.State() or Counted Option.State() [Note: Causes an error when generating logic.]
Option.Quantity() or Counted Option.Quantity()
Optional BOM Model.MinInstances()
Optional NonBOM Model.MinInstances()
Integer or Decimal Feature
Integer or Decimal Feature.Value()
Total or Resource
Total.Value() or Resource.Value()

The value of Second Operand in a Numeric Rule is Decimal and mutable.

C.1.4 Comparison Rule: First Operand

The following nodes and System Properties are valid participants on the First Operand side of a Comparison Rule:

any_node.Property(*true/false*): Converted to a numeric value
any_node.Property(*numeric*)
Required, Optional, or Multiple BOM Model
Required, Optional, or Multiple BOM Model.Selection() [Note: Mutually exclusive only.]
Required, Optional, or Multiple BOM Model.State() [Note: Converted to a numeric value.]
Required or Optional BOM Model.Quantity()
Required or Optional BOM Model.MinInstances()
Required or Optional BOM Model.MaxInstances()
Required or Optional BOM Model.InstanceCount()
Option Class
Option Class.Selection()
Option Class.State() [Note: Converted to a numeric value.]
Option Class.Quantity()
Standard Item
Standard Item.State() [Note: Converted to a numeric value.]
Standard Item.Quantity()
Optional NonBOM Model.MinInstances()
Optional NonBOM Model.MaxInstances()
Optional or Multiple NonBOM Model.InstanceCount()
Multiple NonBOM Model.MinInstances()
Multiple NonBOM Model.MaxInstances()
Optional Component.MinInstances()
Optional Component.MaxInstances()
Optional or Multiple Component Model.InstanceCount()
Multiple Component.MinInstances()
Multiple Component.MaxInstances()
Boolean or Option Feature [Note: Converted to a numeric value.]
Boolean Feature.State() or Count Feature.State() [Note: Converted to a numeric value.]
Count Feature
Count Feature.Quantity()
Option Feature.Selection()
Option Feature.State() [Note: Converted to a numeric value.]
Option [Note: Converted to a numeric value.]
Counted Option
Option.State() or Counted Option.State() [Note: Converted to a numeric value.]
Option.Quantity()or Counted Option.Quantity()
Integer or Decimal Feature

Integer or Decimal Feature.Value()
Total or Resource
Total.Value() or Resource.Value()
Constant

The value of First Operand in a Comparison Rule is Decimal.

C.1.5 Comparison Rule: Second Operand

The following nodes and System Properties are valid participants on the Second Operand side of a Comparison Rule:

any_node.Property(*true/false*)
Required, Optional, or Multiple BOM Model
Required, Optional, or Multiple BOM Model.Options()
Required, Optional, or Multiple BOM Model.Selection(): Mutually exclusive only.
Required, Optional, or Multiple BOM Model.State()
Option Class
Option Class.Options()
Option Class.Selection()
Option Class.State()
Standard Item
Standard Item.State()
Boolean or Option Feature
Boolean Feature.State() or Count Feature.State()
Count Feature
Option Feature.Options()
Option Feature.Selection()
Option Feature.State()
Option
Counted Option
Option.State() or Counted Option.State()

The value of Second Operand in a Comparison Rule is Boolean or a Boolean Collection.

Glossary

This glossary contains definitions that you may need while working with Oracle Configurator.

API

Application Programming Interface

applet

A Java application running inside a Web browser. *See also* **Java** and **servlet**.

Archive Path

The ordered sequence of **Configurator Extension Archive**s for a **Model** that determines which **Java class**es are loaded for **Configurator Extension**s and in what order.

argument

A data value or object that is passed to a method or a **Java class** so that the method can operate.

ATO

Assemble to Order

ATP

Available to Promise

base node

The **node** in a **Model** that is associated with a **Configurator Extension** Rule. Used to determine the **event** scope for a **Configurator Extension**.

bill of material

A list of Items associated with a parent Item, such as an assembly, and information about how each Item relates to that parent Item.

Bills of Material

The application in Oracle Applications in which you define a **bill of material**.

binding

Part of a **Configurator Extension** Rule that associates a specified event with a chosen **method** of a **Java class**. *See also* **event**.

BOM

See **bill of material**.

BOM item

The **node** imported into **Oracle Configurator Developer** that corresponds to an Oracle **Bills of Material** item. Can be a **BOM Model**, **BOM Option Class node**, or **BOM Standard Item node**.

BOM Model

A model that you import from Oracle **Bills of Material** into **Oracle Configurator Developer**. When you import a BOM Model, effective dates, **ATO** rules, and other data are also imported into Configurator Developer. In Configurator Developer, you can extend the structure of the BOM Model, but you cannot modify the BOM Model itself or any of its attribute**s**.

BOM Model node

The imported **node** in **Oracle Configurator Developer** that corresponds to a **BOM Model** created in Oracle **Bills of Material**.

BOM Option Class node

The imported **node** in **Oracle Configurator Developer** that corresponds to a BOM Option Class created in Oracle **Bills of Material**.

BOM Standard Item node

The imported **node** in **Oracle Configurator Developer** that corresponds to a BOM Standard Item created in Oracle **Bills of Material**.

Boolean Feature

An **element** of a **component** in the **Model** that has two **option**s: true or false.

bug

See **defect**.

build

A specific **instance** of an application during its construction. A build must have an install program early in the project so that application **implementer**s can **unit test** their latest work in the context of the entire available application.

CDL

See **Constraint Definition Language**.

CIO

See **Oracle Configuration Interface Object (CIO)**.

command event

An **event** that is defined by a character string, which is considered the command for which **listener**s are listening.

Comparison Rule

An **Oracle Configurator Developer** rule type that establishes a relationship to determine the selection state of a logical **Item** (Option, Boolean Feature, or List-of-Options Feature) based on a comparison of two numeric values (numeric **Feature**s, **Total**s, **Resource**s, **Option** counts, or numeric constants). The numeric values being compared can be computed or they can be discrete intervals in a continuous numeric input.

Compatibility Rule

An **Oracle Configurator Developer** rule type that establishes a relationship among **Features** in the Model to control the allowable combinations of **Options**. *See also*, **Property-based Compatibility Rule**.

Compatibility Table

A kind of Explicit Compatibility Rule. For example, a type of compatibility relationship where the allowable combination of **Options** are explicitly enumerated.

component

A piece of something or a configurable element in a **model** such as a **BOM Model**, **Model**, or **Component**.

Component

An element of the **model structure**, typically containing **Features**, that is configurable and instantiable. An **Oracle Configurator Developer** node type that represents a configurable element of a **Model**. Corresponds to one UI screen of selections in a runtime **Oracle Configurator**.

Component Set

An element of the **Model** that contains a number of instantiated **Components** of the same type, where each Component of the set is independently configured.

concurrent program

Executable code (usually written in SQL*Plus or Pro*C) that performs the function(s) of a requested task. Concurrent programs are stored procedures that perform actions such as generating reports and copying data to and from a database.

configuration

A specific set of specifications for a product, resulting from selections made in a runtime **configurator**.

configuration attribute

A characteristic of an **item** that is defined in the **host application** (outside of its inventory of items), in the **Model**, or captured during a **configuration session**. Configuration attributes are inputs from or outputs to the host application at initialization and termination of the configuration session, respectively.

configuration engine

The part of the runtime **Oracle Configurator** that uses **configuration rules** to validate a **configuration**. Compare **generated logic**.

Configuration Interface Object

See **Oracle Configuration Interface Object (CIO)**.

configuration model

Represents all possible configurations of the available **options**, and consists of **model structure** and **rules**. It also commonly includes **User Interface** definitions and **Configurator Extensions**. A configuration model is usually accessed in a **runtime Oracle Configurator window**. *See also* **model**.

configuration rule

A **Logic Rule**, **Compatibility Rule**, **Comparison Rule**, **Numeric Rule**, **Design Chart**, **Statement Rule**, or **Configurator Extension** rule available in **Oracle Configurator Developer** for defining **configurations**. *See also* **rules**.

configuration session

The time from launching or invoking to exiting **Oracle Configurator**, during which **end users** make selections to configure an orderable product. A configuration session is limited to one **configuration model** that is loaded when the session is initialized.

configurator

The part of an application that provides custom configuration capabilities. Commonly, a window that can be launched from a host application so **end users** can make selections resulting in valid **configurations**. *Compare* **Oracle Configurator**.

Configurator Extension

An extension to the **configuration model** beyond what can be implemented in Configurator Developer.

A type of **configuration rule** that associates a **node**, **Java class**, and event **binding** so that the rule operates when an **event** occurs during a **configuration session**.

A **Java class** that provides methods that can be used to perform configuration actions.

Configurator Extension Archive

An **object** in the **Repository** that stores one or more compiled **Java class**es that implement **Configurator Extension**s.

connectivity

The connection between client and database that allows data communication.

The connection across components of a model that allows modeling such products as networks and material processing systems.

Connector

The **node** in the **model structure** that enables an **end user** at **runtime** to connect the Connector node's parent to a referenced **Model**.

Constraint Definition Language

A language for entering **configuration rule**s as text rather than assembling them interactively in Oracle Configurator Developer. CDL can express more complex constraining relationships than interactively defined configuration rules can.

Container Model

A type of **BOM Model** that you import from Oracle **Bills of Material** into **Oracle Configurator Developer** to create configuration models containing **connectivity** and trackable components. Configurations created from Container Models can be tracked and updated in Oracle Install Base

Contributes to

A relation used to create a specific type of **Numeric Rule** that accumulates a total value. *See also* **Total**.

Consumes from

A relation used to create a specific type of **Numeric Rule** that decrements a total value, such as specifying the quantity of a **Resource** used.

count

The number or quantity of something, such as selected **options**. *Compare* **instance**.

CTO

Configure to Order

customer

The person for whom products are configured by **end users** of the **Oracle Configurator** or other **ERP** and CRM applications. Also the end users themselves directly accessing **Oracle Configurator** in a Web store or kiosk.

customer requirements

The needs of the customer that serve as the basis for determining the configuration of products, **systems**, and services. Also called needs assessment. *See* **guided buying or selling**.

CZ

The product shortname for **Oracle Configurator** in Oracle Applications.

CZ schema

The implementation version of the standard runtime **Oracle Configurator** data-warehousing schema that manages data for the **configuration model**. The implementation schema includes all the data required for the **runtime** system, as well as specific tables used during the construction of the **configurator**.

data import

Populating the **CZ schema** with enterprise data from **ERP** or legacy systems via **import tables**.

data source

A programmatic reference to a database. Referred to by a data source name (DSN).

DBMS

Database Management System

default

A predefined value. In a **configuration**, the automatic selection of an **option** based on the **preselection** rules or the selection of another option.

Defaults relation

An **Oracle Configurator Developer** Logic Rule relation that determines the logic state of **Features** or **Options** in a default relation to other Features and Options. For example, if A Defaults B, and you select A, B becomes Logic True (selected) if it is available (not Logic False).

defect

A failure in a product to satisfy the **users'** requirements. Defects are prioritized as critical, major, or minor, and fixes range from corrections or workarounds to enhancements. Also known as a bug.

Design Chart

An **Oracle Configurator Developer** rule type for defining advanced Explicit Compatibilities interactively in a table view.

developer

The person who uses **Oracle Configurator Developer** to create a **configurator**. *See also* **implementer** and **user**.

Developer

The tool (**Oracle Configurator Developer**) used to create **configuration models**.

DHTML

Dynamic Hypertext Markup Language

discontinued item

A discontinued item is one that exists in an installed configuration of a component (as recorded in Oracle Install Base), but has been removed from the instance of the component being reconfigured, either by deletion or by deselection.

element

Any entity within a **model**, such as **Options**, **Totals**, **Resources**, UI controls, and **components**.

end user

The ultimate user of the runtime **Oracle Configurator**. The types of end users vary by project but may include salespeople or distributors, administrative office staff, marketing personnel, order entry personnel, product engineers, or customers directly accessing the application via a Web browser or kiosk. *Compare* **user**.

enterprise

The **systems** and **resources** of a business.

environment

The arena in which software tools are used, such as operating system, applications, and **server** processes.

ERP

Enterprise Resource Planning. A software system and process that provides automation for the customer's back-room operations, including order processing.

event

An action or condition that occurs in a **configuration session** and can be detected by a **listener**. Example events are a change in the value of a **node**, the creation of a component **instance**, or the saving of a **configuration**. The part of **model structure** inside which a **listener** listens for an event is called the event **binding** scope. The part of model structure that is the source of an event is called the event execution scope. *See also* **command event**.

Excludes relation

An **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Features** or **Options** in an excluding relation to other Features and Options. For example, if A Excludes B, and if you select A, B becomes Logic False, since it is not allowed when A is true (either User or Logic True). If you deselect A (set to User

False), there is no effect on B, meaning it could be User or Logic True, User or Logic False, or **Unknown**. *See* **Negates relation**.

feature

A characteristic of something, or a configurable element of a **component** at **runtime**.

Feature

An element of the **model structure**. Features can either have a value (numeric or Boolean) or enumerated **Options**.

functional specification

Document describing the functionality of the application based on **user** requirements.

generated logic

The compiled structure and rules of a **configuration model** that is loaded into memory on the Web server at **configuration session** initialization and used by the **Oracle Configurator engine** to validate runtime selections. The logic must be generated either in **Oracle Configurator Developer** or programmatically in order to access the configuration model at **runtime**.

guided buying or selling

Needs assessment questions in the **runtime** UI to guide and facilitate the configuration process. Also, the **model structure** that defines these questions. Typically, guided selling questions trigger **configuration rule** that automatically select some product **options** and exclude others based on the **end user's** responses.

host application

An application within which **Oracle Configurator** is embedded as integrated functionality, such as Order Management or *i*Store.

HTML

Hypertext Markup Language

implementation

The stage in a project between defining the problem by selecting a configuration technology vendor, such as Oracle, and deploying the completed configuration application. The implementation stage includes gathering requirements, defining test cases, designing the application, constructing and testing the application, and delivering it to **end users**. *See also* **developer** and **user**.

implementer

The person who uses **Oracle Configurator Developer** to build the **model structure**, **rules**, and UI customizations that make up a **runtime** Oracle Configurator. Commonly also responsible for enabling the integration of **Oracle Configurator** in a **host application**.

Implies relation

An **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Features** or **Options** in an implied relation to other Features and Options. For example, if A Implies B, and you select A, B becomes Logic True. If you deselect A (set to User False), there is no effect on B, meaning it could be User or Logic True, User or Logic False, or **Unknown**. *See* **Requires relation**.

import server

A database **instance** that serves as a source of data for **Oracle Configurator**'s Populate, Refresh, and Synchronization concurrent processes. The import server is sometimes referred to as the remote server.

import tables

Tables mirroring the CZ schemaItem Master structure, but without integrity constraints. Import tables allow batch population of the CZ schema's Item Master. Import tables also store extractions from Oracle Applications or **legacy data** that create, update, or delete records in the CZ schema **Item Master**.

initialization message

The **XML** message sent from a **host application** to the **Oracle Configurator Servlet**, containing data needed to initialize the runtime Oracle Configurator. *See also* **termination message**.

Instance

An **Oracle Configurator Developer** attribute of a **component's node** that specifies a minimum and maximum value. *See also* **instance**.

instance

A **runtime** occurrence of a **component** in a configuration. *See also* **instantiate**. *Compare* **count**.

Also, the memory and processes of a database.

instantiate

To create an instance of something. Commonly, to create an **instance** of a **component** in the runtime **user interface** of a **configuration model**.

integration

The process of combining multiple software **components** and making them work together.

integration testing

Testing the interaction among software programs that have been integrated into an application or **system**. Also called system testing. *Compare* **unit test**.

item

A product or part of a product that is in inventory and can be delivered to customers.

Item

A Model or part of a Model that is defined in the **Item Master**. Also data defined in Oracle Inventory.

Item Master

Data stored to structure the Model. Data in the **CZ schema** Item Master is either entered manually in **Oracle Configurator Developer** or imported from Oracle Applications or a legacy system.

Item Type

Data used to classify the Items in the Item Master. Item Catalogs imported from Oracle Inventory are Item Types in **Oracle Configurator Developer**.

Java

An object-oriented programming language commonly used in internet applications, where Java applications run inside Web browsers and **servers**. Used to implement the behavior of **Configurator Extension**s. *See also* **applet** and **servlet**.

Java class

The compiled version of a **Java** source code file. The **method**s of a Java class are used to implement the behavior of **Configurator Extension**s.

JavaServer Pages

Web pages that combine static presentation elements with dynamic content that is rendered by Java **servlet**s.

JSP

See **JavaServer Pages**.

legacy data

Data that cannot be imported without creating custom extraction programs.

listener

A class in the **CIO** that detects the occurrence of specified **event**s in a **configuration session**.

load

Storing the **configuration model** data in the **Oracle Configurator Servlet** on the Web server. Also, the time it takes to initialize and display a configuration model if it is not preloaded.

The burden of transactions on a **system**, commonly caused by the ratio of **user** connections to CPUs or available memory.

log file

A file containing errors, warnings, and other information that is output by the running application.

Logic Rule

An **Oracle Configurator Developer** rule type that expresses constraint among model elements in terms of logic relationships. Logic Rules directly or indirectly set the logical state (User or Logic True, User or Logic False, or **Unknown**) of **Features** and **Options** in the Model.

There are four primary Logic Rule relations: Implies, Requires, Excludes, and Negates. Each of these rules takes a list of Features or Options as operands. *See also* **Implies relation**, **Requires relation**, **Excludes relation**, and **Negates relation**.

maintainability

The characteristic of a product or process to allow straightforward **maintenance**, alteration, and extension. Maintainability must be built into the product or process from inception.

maintenance

The effort of keeping a **system** running once it has been deployed, through **defect** fixes, procedure changes, infrastructure adjustments, data replication schedules, and so on.

Metalink

Oracle's technical support Web site at:

`http://www.oracle.com/support/metalink/`

method

A function that is defined in a **Java class**. Methods perform some action and often accept parameters.

Model

The entire hierarchical "tree" view of all the data required for **configurations**, including **model structure**, variables such as **Resources** and **Totals**, and elements in support of intermediary rules. Includes both imported **BOM Models** and Models created in Configurator Developer. May consist of BOM Option Classes and BOM Standard Items.

model

A generic term for data representing products. A model contains **elements** that correspond to **items**. Elements may be **components** of other objects used to define products. A **configuration model** is a specific kind of model whose elements can be configured by accessing an **Oracle Configurator window**.

model-driven UI

The graphical views of the **model structure** and **rules** generated by **Oracle Configurator Developer** to present **end users** with interactive product selection based on **configuration models**.

model structure

Hierarchical "tree" view of data composed of **elements** (**Models**, **Components**, **Features**, **Options**, **BOM Models**, **BOM Option Class nodes**, **BOM Standard Item nodes**, **Resources**, and **Totals**). May include reusable **components** (**References**).

Negates relation

A type of **Oracle Configurator Developer Logic Rule** type that determines the logic state of **Features** or **Options** in a negating relation to other Features and Options. For example, if one **option** in the relationship is selected, the other option must be Logic False (not selected). Similarly, if you deselect one option in the relationship, the other option must be Logic True (selected). *See* **Excludes relation**.

node

The icon or location in a **Model** tree in **Oracle Configurator Developer** that represents a **Component**, **Feature**, **Option** or variable (**Total** or **Resource**), **Connector**, **Reference**, **BOM Model**, **BOM Option Class node**, or **BOM Standard Item node**.

Numeric Rule

An **Oracle Configurator Developer** rule type that expresses constraint among model elements in terms of numeric relationships. *See also*, **Contributes to** and **Consumes from**.

object

Entities in **Oracle Configurator Developer**, such as **Models**, Usages, Properties, Effectivity Sets, UI Templates, and so on. *See also* **element**.

OC

See **Oracle Configurator**.

OCD

See **Oracle Configurator Developer**.

option

A logical selection made in the Model Debugger or a runtime Oracle Configurator by the **end user** or a rule when configuring a **component**.

Option

An element of the **Model**. A choice for the value of an enumerated **Feature**.

Oracle Configuration Interface Object (CIO)

A **server** in the **runtime** application that creates and manages the interface between the client (usually a **user interface**) and the underlying representation of **model structure** and **rules** in the **generated logic**.

The CIO is the **API** that supports creating and navigating the Model, querying and modifying selection states, and saving and restoring **configurations**.

Oracle Configurator

The product consisting of development tools and **runtime** applications such as the **CZ schema**, **Oracle Configurator Developer**, and runtime Oracle Configurator. Also the runtime Oracle Configurator variously packaged for use in networked or Web deployments.

Oracle Configurator architecture

The three-tier **runtime** architecture consists of the **User Interface**, the **generated logic**, and the **CZ schema**. The application development architecture consists of **Oracle Configurator Developer** and the CZ schema, with test instances of a runtime **Oracle Configurator**.

Oracle Configurator Developer

The suite of tools in the **Oracle Configurator** product for constructing and maintaining **configurator**s.

Oracle Configurator engine

The part of the **Oracle Configurator** product that validates runtime selections. *See also* **generated logic**.

Oracle Configurator schema

See **CZ schema**.

Oracle Configurator Servlet

A **Java** servlet that participates in rendering Legacy user interfaces for **Oracle Configurator**.

Oracle Configurator window

The **user interface** that is launched by accessing a **configuration model** and used by **end users** to make the selections of a **configuration**.

performance

The operation of a product, measured in throughput and other data.

Populator

An entity in **Oracle Configurator Developer** that creates **Component**, **Feature**, and **Option nodes** from information in the **Item Master**.

preselection

The default state in a **configurator** that defines an initial selection of **Components**, **Features**, and **Options** for configuration.

A process that is implemented to select the initial element(s) of the **configuration**.

product

Whatever is ordered and delivered to customers, such as the output of having configured something based on a model. Products include intangible entities such as services or contracts.

Property

A named value associated with a **node** in the **Model** or the **Item Master**. A set of Properties may be associated with an Item Type. After importing a BOM Model, Oracle Inventory Catalog Descriptive Elements are Properties in **Oracle Configurator Developer**.

Property-based Compatibility Rule

An **Oracle Configurator Developer** Compatibility Rule type that expresses a kind of compatibility relationship where the allowable combinations of **Options** are specified implicitly by relationships among Property values of the Options.

prototype

A construction technique in which a preliminary version of the application, or part of the application, is built to facilitate **user** feedback, prove feasibility, or examine other implementation issues.

PTO

Pick to Order

publication

A unique deployment of a **configuration model** (and optionally a **user interface**) that enables a developer to control its availability from host applications such as Oracle Order Management or *i*Store. Multiple publications can exist for the same configuration model, but each publication corresponds to only one **Model** and **User Interface**.

publishing

The process of creating a **publication** record in **Oracle Configurator Developer**, which includes specifying applicability parameters to control **runtime** availability and running an Oracle Applications concurrent process to copy data to a specific database.

RDBMS

Relational Database Management System

reference

The ability to reuse an existing **Model** or **Component** within the structure of another Model (for example, as a subassembly).

Reference

An **Oracle Configurator Developer** node type that denotes a **reference** to another **Model**.

Repository

Set of pages in **Oracle Configurator Developer** that contains areas for organizing and maintaining **Model**s and shared **object**s in a single location.

Requires relation

An **Oracle Configurator Developer** Logic Rule relationship that determines the logic state of **Features** or **Options** in a requirement relation to other Features and Options. For example, if A Requires B, and if you select A, B is set to Logic True (selected). Similarly, if you deselect A, B is set to Logic False (deselected). See **Implies relation**.

resource

Staff or equipment available or needed within an enterprise.

Resource

A variable in the **Model** used to keep track of a quantity or supply, such as the amount of memory in a computer. The value of a Resource can be positive or zero, and can have an Initial Value setting. An error message appears at **runtime** when the value of a Resource becomes negative, which indicates it has been over-consumed. Use **Numeric Rules** to contribute to and consume from a Resource.

Also a specific node type in **Oracle Configurator Developer**. *See also* **node**.

reusable component

See **reference** and **model structure**.

reusability

The extent to and ease with which parts of a **system** can be put to use in other systems.

rules

Also called business rules or **configuration rule**. In the context of Oracle Configurator and **CDL**, a rule is not a "business rule." Constraints applied among elements of the product to ensure that defined relationships are preserved during configuration. Elements of the product are **Components**, **Features**, and **Options**. Rules express logic, numeric parameters, implicit compatibility, or explicit compatibility. Rules provide **preselection** and **validation** capability in **Oracle Configurator**.

See also **Comparison Rule**, **Compatibility Rule**, **Design Chart**, **Logic Rule** and **Numeric Rule**.

runtime

The environment and context in which applications are run, tested, or used, rather than developed.

The environment in which an **implementer** (tester), **end user**, or **customer** configures a product whose model was developed in **Oracle Configurator Developer**. *See also* **configuration session**.

schema

The tables and objects of a data model that serve a particular product or business process. *See also* **CZ schema**.

server

Centrally located software processes or hardware, shared by client**s**.

servlet

A Java application running inside a Web server. *See also* **Java**, **applet**, and **Oracle Configurator Servlet**.

solution

The deployed **system** as a response to a problem or problems.

SQL

Structured Query Language

Statement Rule

An **Oracle Configurator Developer** rule type defined by using the Oracle Configurator **Constraint Definition Language** (text) rather than interactively assembling the rule's elements.

system

The hardware and software **component**s and infrastructure integrated to satisfy functional and **performance** requirements.

termination message

The **XML** message sent from the **Oracle Configurator Servlet** to a **host application** after a **configuration session**, containing configuration outputs. *See also* **initialization message**.

Total

A variable in the **Model** used to accumulate a numeric total, such as total price or total weight.

Also a specific node type in **Oracle Configurator Developer**. *See also* **node**.

UI

See **User Interface**.

UI Templates

Templates available in **Oracle Configurator Developer** for specifying UI definitions.

Unknown

The logic state that is neither true nor false, but unknown at the time a **configuration session** begins or when a Logic Rule is executed. This logic state is also referred to as Available, especially when considered from the point of view of the **runtime Oracle Configurator end user**.

unit test

Execution of individual routines and modules by the application **implementer** or by an independent test consultant to find and resolve **defect**s in the application. *Compare* **integration testing**.

update

Moving to a new version of something, independent of software release. For instance, moving a production **configurator** to a new version of a **configuration model**, or changing a **configuration** independent of a model **update**.

upgrade

Moving to a new release of **Oracle Configurator** or **Oracle Configurator Developer**.

user

The person using a product or system. Used to describe the person using **Oracle Configurator Developer** tools and methods to build a **runtime Oracle Configurator**. *Compare* **end user**.

User Interface

The part of an **Oracle Configurator** implementation that provides the graphical views necessary to create **configuration**s interactively. A **user interface** is generated from the **model structure**. It interacts with the model definition and the **generated logic** to give **end user**s access to customer requirements gathering, product selection, and any extensions that may have been implemented. *See also* **UI Templates**.

user interface

The visible part of the application, including menus, dialog boxes, and other on-screen elements. The part of a **system** where the **user** interacts with the software. Not necessarily generated in **Oracle Configurator Developer**. *See also* **User Interface**.

user requirements

A description of what the **configurator** is expected to do from the **end user's** perspective.

validation

Tests that ensure that configured **components** will meet specific criteria set by an enterprise, such as that the components can be ordered or manufactured.

variable

Parts of the **Model** that are represented by **Totals**, **Resources**, or numeric **Features**.

verification

Tests that check whether the result agrees with the specification.

Web

The portion of the Internet that is the World Wide Web.

Workbench

Set of pages in **Oracle Configurator Developer** for creating, editing, and working with **Repository object**s such as **Model**s and **UI Templates**.

XML

Extensible Markup Language, a highly flexible markup language for transferring data between **Web** applications. Used for the **initialization message** and **termination message** of the **Oracle Configurator Servlet**.

Index