

Oracle® Configurator

Oracle Configuration Interface Object (CIO) Developer's Guide

Release 11*i*

September 2000

Part No. A81001-04

This document describes Functional Companions, which augment the functionality of an Oracle SellingPoint application, and the Oracle Configuration Interface Object (CIO), which is used by Functional Companions to access the Oracle Configurator Active Model.

Part No. A81001-04

Copyright © 1996, 2000, Oracle Corporation. All rights reserved.

Primary Author: Mark Sawtelle

Contributors: Brent Benson, Jim Carlson, Ivan Lazarov, Marty Plotkin, Brian Ross

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

Program Documentation is licensed for use solely to support the deployment of the Programs and not for any other purpose.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and *Oracle SellingPoint Configurator* is a trademark or registered trademark of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Contents

Send Us Your Comments	xi
Preface.....	xiii
Intended Audience	xiii
Structure.....	xiii
Related Documents.....	xiv
Conventions.....	xiv
 1 Functional Companions	
1.1 What Are Functional Companions?	1-1
1.1.1 Types of Functional Companions.....	1-1
1.1.2 Background to Building Functional Companions.....	1-2
1.2 Functional Companions and the CIO.....	1-3
1.2.1 Using the CIO Interface.....	1-4
1.2.2 Implementing Standard Interface Methods	1-4
1.3 Building Functional Companions in Java.....	1-4
1.3.1 Procedure for Building Functional Companions in Java	1-4
1.3.2 Installation Requirements for Java Functional Companions.....	1-7
1.3.2.1 Requirements for Developing Functional Companions	1-7
1.3.2.2 Requirements for Running Functional Companions.....	1-7
1.3.2.3 Requirements for Testing Java Functional Companions	1-8
1.3.3 Minimal Example of a Java Functional Companion	1-9
1.4 Incorporating Functional Companions in your Application.....	1-10

1.4.1	Associating Functional Companions with your Model.....	1-10
1.4.2	Testing Functional Companions in the Oracle SellingPoint Application	1-14
1.4.2.1	Testing from the Windows Start Menu.....	1-15
1.4.2.2	Testing from Oracle Configurator Developer	1-15
1.4.2.3	Test Functionality in the Oracle SellingPoint Application.....	1-15

2 The Configuration Interface Object (CIO)

2.1	Background	2-1
2.1.1	What is the CIO?	2-1
2.1.2	The CIO and Functional Companions.....	2-2
2.2	The CIO's Runtime Node Interface Classes.....	2-2
2.3	Initializing the CIO	2-4
2.4	Access to Configurations.....	2-5
2.4.1	Creating and Deleting Configurations	2-6
2.4.2	Saving and Restoring Configurations	2-7
2.4.3	Access to Configuration Parameters	2-7
2.4.4	Logic Transactions.....	2-8
2.5	Access to Nodes of the Model at Runtime.....	2-9
2.5.1	Opportunities for Modifying the Model.....	2-9
2.5.2	Accessing Components	2-10
2.5.3	Adding and Deleting Optional Components.....	2-10
2.5.4	Accessing Features	2-10
2.5.5	Getting and Setting Logic States	2-11
2.5.6	Getting and Setting Numeric Values.....	2-13
2.5.7	Accessing Properties	2-14
2.5.8	Access to Options	2-14
2.6	Introspection through IRuntimeNode.....	2-15
2.7	Handling Logical Contradictions.....	2-18
2.7.1	Generating Error Messages from Contradictions.....	2-18
2.7.2	Raising Exceptions	2-19
2.7.3	Overriding Contradictions.....	2-20
2.8	Validating Configurations.....	2-21
2.9	Standard Interface Methods for Functional Companions	2-22
2.9.1	The initialize() Interface Method.....	2-23
2.9.2	The autoConfigure() Interface Method	2-25

2.9.3	The validate() Interface Method.....	2–26
2.9.4	The generateOutput() Interface Method.....	2–27
2.9.5	The terminate() Interface Method.....	2–28

3 Reference Documentation for the CIO

4 Examples

4.1	Initializing the CIO.....	4–1
4.2	Basic Java Functional Companion	4–2
4.3	Thin-Client generateOutput() Functional Companion.....	4–8

Glossary of Terms

Glossary of Acronyms

A CIO Package and Related Classes

B Package oracle.apps.cz.cio

C Package oracle.apps.cz.common

D Package oracle.apps.cz.utilities

Index

List of Examples

1-1	Elementary Java Functional Companion: MyClass.java	1-9
2-1	Initializing the CIO (Short Example)	2-5
2-2	Creating New Configuration Objects	2-6
2-3	Getting the state of a node	2-12
2-4	Setting the state of a node	2-12
2-5	Setting a numeric value	2-13
2-6	Testing whether an option is selected, or satisfied	2-15
2-7	Testing whether a node is selected, or satisfied	2-17
2-8	Getting a child node by name	2-17
2-9	Collecting all child nodes by type	2-18
2-10	Handling and overriding Logical Exceptions	2-20
4-1	Initializing the CIO (Long Example)	4-1
4-2	Basic Functional Companion: FuncCompTest1	4-2
4-3	Thin-client Output Functional Companion	4-10

List of Figures

1-1	Associating a Component with a Functional Companion	1-13
1-2	Functional Companion Rule: Detail of the Attributes view.....	1-14
1-3	Testing Functional Companions in the Oracle SellingPoint application.	1-16
1-4	Modifying Functional Companion Buttons.....	1-17

List of Tables

1-1	Types of Functional Companions	1-2
1-2	Required Software for Functional Companions	1-8
2-1	Runtime node interface classes for the CIO	2-2
2-2	Methods of the Interface Class IOption	2-15
2-3	Methods of the interface class IRuntimeNode	2-15
2-4	Standard methods of the IFunctionalCompanion interface.....	2-23

Send Us Your Comments

Oracle Configuration Interface Object (CIO) Developer's Guide, Release 11*i*

Part No. A81001-04

Oracle Corporation 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 chapter, section, and page number (if available). You can send comments through your call to Oracle Support Services or by sending them to:

Oracle Configurator
Oracle Corporation
Documentation
21 North Avenue
Burlington, MA 01803
USA

If you would like a reply, please give your name, address, and telephone number below.

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

Preface

You can use Functional Companions to augment the functionality of your Oracle SellingPoint application beyond what is provided by Oracle Configurator Developer. You create Functional Companion objects, which use the Configuration Interface Object (CIO) to perform various tasks, including accessing the Model, setting and getting logic states, and adding optional components. You can also use the CIO in your own applications, to interact with the Model.

Intended Audience

This manual is intended primarily for software developers writing Functional Companions. The language recommended for developing Functional Companions is Java.

This manual assumes that you are an experienced programmer and that you understand Oracle databases, the SQL and Java programming languages, and the principles of JDBC.

This manual also provides background and reference information on the CIO, which is needed by developers of applications having customized user interfaces that need access to the Oracle Configurator Active Model.

Structure

This manual contains:

- [Chapter 1, "Functional Companions"](#)
- [Chapter 2, "The Configuration Interface Object \(CIO\)"](#)

- [Chapter 3, "Reference Documentation for the CIO"](#)
- [Chapter 4, "Examples"](#)

Related Documents

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

- *Oracle SellingPoint Configurator Administration Guide*
- *Oracle Configurator Developer User's Guide*
- *Oracle Configurator Developer Tutorial*

The following documents are also relevant:

- *Oracle8i JDBC Developer's Guide and Reference*

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 following conventions 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.
. . .	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 term defined in the text, the glossary, or in both locations.
< >	Angle brackets enclose user-supplied names.
[]	Brackets enclose optional clauses from which you can choose one or none.
>	The left bracket alone sign represents the MS DOS prompt.

Functional Companions

Functional Companions extend your Oracle SellingPoint application by attaching custom code through established interfaces.

1.1 What Are Functional Companions?

A Functional Companion is a programming object that you attach to your Model in order to extend the functionality of your Oracle SellingPoint application in ways that are not provided by Oracle Configurator Developer.

You can write a Functional Companion object in several languages, depending on the functionality needed by your application. The Functional Companion communicates with your Model through an API (application programming interface) called the Configuration Interface Object (CIO). The Oracle Configuration Interface Object is written in Java. See [Chapter 2, "The Configuration Interface Object \(CIO\)"](#).

You connect Functional Companions to specific nodes in your Model using Oracle Configurator Developer. You also specify the type of action that you want the specified Functional Companion to perform when your end users select its associated node. Then you generate the logic and user interface, as you normally do for your Oracle SellingPoint application. This action associates the Functional Companion with your application so that when your end users select a node in the Model, the Functional Companion on that node is automatically invoked.

1.1.1 Types of Functional Companions

You can assign a Functional Companion to perform any or all of these three types of actions:

Table 1–1 Types of Functional Companions

Type	Description
Auto-configuration	Configures the state of the Model. You can use this to modify the shape of the Model tree, and the state of its nodes. For instance, your application might gather initial needs assessment information and use it to set up the appropriate set of choices for your end user to make. In your Oracle SellingPoint application, your end user will explicitly choose to run an auto-configuration Functional Companion See Section 2.9.2, "The autoConfigure() Interface Method".
Validation	Validates the logical choice that the end user has just made. The Functional Companion can perform complex operations beyond the scope of what you can develop in Oracle Configurator Developer. For instance, you can perform sophisticated numeric comparisons. A Java Functional Companion returns null if the validation is successful. If the validation fails, it returns a List of CompanionValidationFailure objects. In your Oracle SellingPoint application, all validation Functional Companions are run every time your end user chooses an Option. After each action, the end user gets the collection of strings returned by each Functional Companion that failed. Validation companions query the Model to determine validity, but should <i>not</i> modify the Model. Modifying the Model in a validation Functional Companion can cause unexpected application failures. See Section 2.9.3, "The validate() Interface Method".
Output	Generates some form of output from the configuration. This output might be a report, a performance graph, a geometric rendering, or a graphical representation of the configuration. In your Oracle SellingPoint application, your end user will explicitly choose to run an output Functional Companion. See Section 2.9.4, "The generateOutput() Interface Method".

1.1.2 Background to Building Functional Companions

To build a Functional Companion, you implement an object class in the language that you choose as being most appropriate for the operation that you want to

perform. Oracle recommends using Java for developing Functional Companions. Java Functional Companions can run on any platform supported by Java.

When an Oracle SellingPoint application runs, it creates an instance of the CIO, which creates runtime instances of all the Components in the Model. If you used Oracle Configurator Developer to associate a Functional Companion with a Component, then the application creates, for each instance of that Component, an instance of the class that you defined for your Functional Companion and attaches the Functional Companion instance to the Component.

You can associate more than one Functional Companion with a particular Component; the CIO will create instances of all of them.

If any Functional Companions cannot be loaded when you create a new configuration (for instance, due to internal errors or an incorrect CLASSPATH), the configuration will fail to open.

You can also associate Functional Companions with Products.

- For Functional Companions built with Java, you implement a class that extends `oracle.apps.cz.cio.FunctionalCompanion`. See [Section 1.3, "Building Functional Companions in Java"](#).
- For *all* language choices, you also implement one or more of the standard interface methods of `oracle.apps.cz.cio.IFunctionalCompanion`, which are described in [Section 2.9, "Standard Interface Methods for Functional Companions"](#).

In order to communicate with the Model of your application, the Functional Companion uses Oracle's CIO API. The CIO can also be used to develop a custom user interface for an Oracle SellingPoint application, in order to access the Model. As a point of information, both Oracle Configurator Developer and the default user interface for the Oracle SellingPoint application communicate in just this way with the Model, using the Oracle Configurator schema to store structure, rules, and user interface information (in addition to your end user's data).

1.2 Functional Companions and the CIO

Functional Companions are invoked by the CIO through the Oracle SellingPoint application, and Functional Companions call the CIO to get information from the Active Model. The CIO is like a broker for the Active Model, in that it passes information both ways. Programmers writing Functional Companions need to know how to use the CIO.

Each Functional Companion is an object class. For every Component instance in your Model that is associated with a Functional Companion, the CIO creates an instance of this class.

1.2.1 Using the CIO Interface

Your Functional Companion is a client of the CIO. When you program against the CIO, you create instances of a set of public interface objects, which are defined in `oracle.apps.cz.cio`.

Your code should refer only to these public interface objects. See [Section 2.2, "The CIO's Runtime Node Interface Classes"](#).

Reference

For reference documentation, see: [Package oracle.apps.cz.cio](#).

1.2.2 Implementing Standard Interface Methods

You provide functionality for your Functional Companion by implementing body code for the methods:

- `initialize`
- `autoConfigure`
- `validate`
- `generateOutput`
- `terminate`

These methods are described in [Section 2.9, "Standard Interface Methods for Functional Companions"](#).

For particulars that apply to the languages currently supported by the CIO, and examples, see [Section 1.3, "Building Functional Companions in Java"](#).

1.3 Building Functional Companions in Java

1.3.1 Procedure for Building Functional Companions in Java

Here is an overview of the tasks for Building Functional Companions in Java. See also [Section 1.3.2, "Installation Requirements for Java Functional Companions"](#).

1. Use a Java development environment or text editor to create a .java file in which to define a Java class.
2. Import the classes for the CIO (`oracle.apps.cz.cio.*`).

```
import oracle.apps.cz.cio.*;
```

3. Define a class in which to determine the behavior of your Functional Companion.

Here is the relevant line from [Example 1-1](#):

```
public class MyClass extends FunctionalCompanion // line 6
```

When you define your Functional Companion class, you can do one of the following:

- Normally: Extend the base class for Functional Companions—`oracle.apps.cz.cio.FunctionalCompanion`—and override just the particular methods that you need. In this case, you gain the functionality of the `FunctionalCompanion` base class. This functionality includes: saving references to the runtime node with which the Functional Companion is associated (with the `FunctionalCompanion.getRuntimeNode()` method), and returning the name of the Functional Companion (with the `FunctionalCompanion.getName()` method). See the reference for: [FunctionalCompanion](#).
 - More rarely: Implement the interface class for Functional Companions—`oracle.apps.cz.cio.IFunctionalCompanion`—and implement all its methods. You do not extend `oracle.apps.cz.cio.FunctionalCompanion`. In this case, you lose the functionality of the `FunctionalCompanion` base class. See the reference for: [FunctionalCompanion](#).
4. You may want to override `oracle.apps.cz.cio.FunctionalCompanion.initialize()`. (See [Section 2.9.1, "The initialize\(\) Interface Method"](#).)

You should ordinarily never directly call `FunctionalCompanion.initialize()`, since the CIO does that for you. However, if your Functional Companion overrides `FunctionalCompanion` as its base class, then the `initialize()` method of *your* class should call `super.initialize()`. This passes some necessary variables to the superclass (`oracle.apps.cz.cio.FunctionalCompanion`) so that its methods will work.

For an example in context, see [Line 35](#) in [Example 4-2, "Basic Functional Companion: FuncCompTest1"](#) on page 4-2, which is shown below:

```
public void initialize(IRuntimeNode comp_node, String name, String
description, int id)
{
    this.comp_node = comp_node;
    super.initialize(comp_node, name, description, id); // line 35
}
```

5. Override one or more of the other interface methods of `oracle.apps.cz.cio.IFunctionalCompanion` (see [Section 2.9, "Standard Interface Methods for Functional Companions"](#)):

```
autoConfigure
validate
generateOutput
terminate
```

For examples in context, see [Example 4-2, "Basic Functional Companion: FuncCompTest1"](#) on page 4-2:

```
public void autoConfigure()

public List validate()

public String generateOutput()
```

6. Optionally, call the methods of the other interface classes of the CIO (see [Section 2.2, "The CIO's Runtime Node Interface Classes"](#)).

Note: Basic Functional Companions, ones that only use the standard interface methods listed in step 5, do not need to use the interface classes of the CIO.

7. Compile the .java file into a .class file for example, with JDK 1.1.x:

```
javac FuncCompTest1.java
```

8. Put the resulting .class file in your classpath, or into a JAR file in your classpath. For example:

```
jar cvf FuncComps.jar FuncCompTest1.class
```

```
set CLASSPATH=%CLASSPATH%;D:\companions\FuncComps.jar
```

9. Run Oracle Configurator Developer with this classpath. Associate your Functional Companion with a Component in your Model. See [Section 1.4, "Incorporating Functional Companions in your Application"](#) on page 1-10. Generate the Active Model and User Interface.
10. To test your Functional Companion, click the Test button in Oracle Configurator Developer. When the Oracle SellingPoint application runs, click the buttons that have been generated in the UI for activating your Functional Companions. See [Section 1.4.2, "Testing Functional Companions in the Oracle SellingPoint Application"](#) on page 1-14.

1.3.2 Installation Requirements for Java Functional Companions

1.3.2.1 Requirements for Developing Functional Companions

In order to develop Java Functional Companions, you must install a Java development environment that enables you to compile Java classes, such as:

- Oracle JDeveloper
- Sun JDK 1.1.x or JDK 1.2.x (JDK 1.1.x is recommended for compatibility with Oracle Applications Release 11i)
- Microsoft Visual J++

You do not need JDBC drivers or database access to compile a Functional Companion, although these are required to run one.

1.3.2.2 Requirements for Running Functional Companions

At runtime, an Oracle SellingPoint application using Functional Companions requires:

- The Microsoft Java Virtual Machine (JVM)
- Microsoft JDBC/ODBC drivers
- An ODBC datasource

The Oracle SellingPoint application automatically sets up a JDBC database connection for use by the CIO. Custom user interfaces that take the place of the Oracle SellingPoint application must perform this task. See [Section 2.3, "Initializing the CIO"](#) for details.

In order to run Java Functional Companions, the software described in [Table 1–2](#) must be installed and recognized by your operating system environment in the indicated locations.

Table 1–2 Required Software for Functional Companions

File name	Location	Required for	Source
config.jar	CLASSPATH	Any use of CIO.	Oracle Configurator (OC) installation.
cz3rdpty.jar	CLASSPATH	Use of the Java classes for collections, the XML parser, the Swing UI, and various fundamentals.	OC installation.
cz.dll czjni.dll	PATH	CIO and Functional Companion access to the Oracle Configurator logic engine.	OC installation.
JDBC OCI driver (such as oci805jdbc.dll)	PATH	Functional Companions using Oracle JDBC OCI (“thick”)drivers. For use with Javasoft JDK 1.1.x.	Oracle Technology Network download area, under “Oracle 8 JDBC OCI and JDBC Thin Drivers”.
classes111.zip	CLASSPATH	Functional Companions using the Oracle JDBC Thin drivers. For use with Javasoft JDK 1.1.x.	Oracle Technology Network download area, under “Oracle 8 JDBC OCI and JDBC Thin Drivers”.

For background on JDBC drivers, see the *Oracle8i JDBC Developer’s Guide and Reference*.

1.3.2.3 Requirements for Testing Java Functional Companions

The class(es) that implement your Functional Companions must be included in your CLASSPATH environment variable. Otherwise, you are likely to get an error message like the following when you try to create a new configuration:

```
New Configuration: Cannot create configuration:
oracle.apps.cz.cio.FuncCompCreationException:
java.lang.ClassNotFoundException: classname
```

Where *classname* is the name of the first Functional Companion to be loaded.

1.3.3 Minimal Example of a Java Functional Companion

[Example 1–1](#) illustrates the minimal coding required for a Functional Companion that does not perform any work. (See [Section 4.2, "Basic Java Functional Companion"](#) for a fuller example.)

Example 1–1 Elementary Java Functional Companion: MyClass.java

```
import oracle.apps.cz.cio.*;
import com.sun.java.util.collections.List;           // line 2
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class MyClass extends FunctionalCompanion     // line 6
{
    // constructor
    public MyClass(CIO cio, IRuntimeNode node) {
    }

    public void initialize(IRuntimeNode node, String name, String description,
int id) {
        // implement body, if necessary
        super.initialize(node, name, description, id);
    }

    public void autoConfigure() throws LogicalException {
        // implement body as desired
    }

    public List validate() {
        // implement body as desired
        return null;
    }

    // for thin client
    public void generateOutput(HttpServletResponse response) throws IOException
    {
        // implement body as desired
    }

    // for thick client
    public String generateOutput() {
        // implement body as desired
    }
}
```

```
        return null;
    }

    public void terminate() {
        // implement body, if necessary
        super.terminate();
    }
}
```

Line 2

```
import com.sun.java.util.collections.List; // line 2
```

If you are using JDK 1.1.x, import `com.sun.java.util.collections.List`, which is provided in `cz3rdpty.jar` (see ["Installation Requirements for Java Functional Companions"](#) on page 1-7). If you are using JDK 1.2, then import `java.util.List`.

Line 4

```
public class MyClass extends FunctionalCompanion // line 6
```

This class extends the base class for Functional Companions: `oracle.apps.cz.cio.FunctionalCompanion`. See the explanation under Step 3.

1.4 Incorporating Functional Companions in your Application

1.4.1 Associating Functional Companions with your Model

To enable your Functional Companion to work with your Oracle SellingPoint application, you must associate it with a Component (or Product) in your Model. You create this association in Oracle Configurator Developer, as a type of Configuration Rule that specifies the Functional Companion method(s) that you have implemented, and the path to be used by the Oracle SellingPoint application to locate the Functional Companion object.

To create an association between a Component and a Functional Companion:

1. Click on the Rules button on the main toolbar.

A list of the Configuration Rule types appear in the lower-left pane.

2. Choose **New Functional Companion** from the Create menu. You can also highlight the Functional Companions node, click on the right mouse button, and select New Functional Companion from the popup menu.
3. Type a name for the Functional Companion rule.
4. In the Description section, type a short explanation of the Functional Companion rule. If necessary, open the Description section by clicking on the blue arrow to the left of it.
5. If necessary, open the Definition section by clicking on the blue arrow to the left of it. In the Model view, select the Component or Product that you want to include in this rule. Drag it with the left-hand mouse button to the Base Component field in the Definition section. Only one Base Component may be specified per rule.
6. Choose one or more roles for the Functional Companion. The choices are:

Type	Associated Functional Companion method
Auto-configuration	autoConfigure
Validation	validate
Output	generateOutput

See [Section 1.1.1, "Types of Functional Companions"](#) and [Section 2.9, "Standard Interface Methods for Functional Companions"](#) for background. Note that you do not associate the `initialize()` and `terminate()` methods, since they are invoked automatically by the Oracle SellingPoint application.

7. Indicate how the Functional Companion is implemented:
 - Java
 - Custom
8. Type in the Program String that identifies the Functional Companion:
 - For Java, this is the name of the class that implements the Functional Companion, such as:

`FuncCompTest1`

The full class specification must be accessible through your CLASSPATH environment variable. For instance, if `FuncCompTest1` is contained this way in a JAR file `tests.jar`:

```
com\java\tests\FuncCompTest1.class
```

then you would specify the Functional Companion this way in Configurator Developer:

```
com.java.tests.FuncCompTest1
```

See Step 8 under [Section 1.3.1, "Procedure for Building Functional Companions in Java"](#).

9. Choose Generate Active Model from the Tools menu.
10. When the Generate Active Model command completes successfully, click on the UI button on the main toolbar, then choose Refresh from the Edit menu.

[Figure 1–1](#) shows what the Rules module screen of Oracle Configurator Developer might look like after you associate a Component with a Functional Companion.

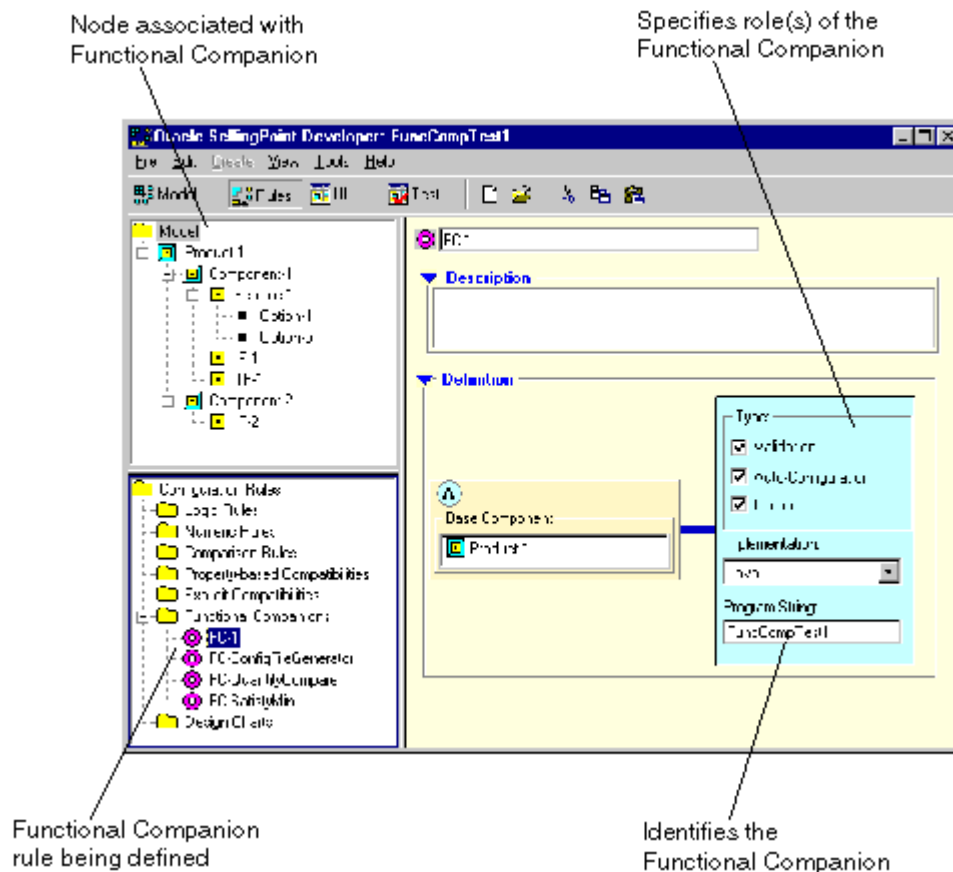
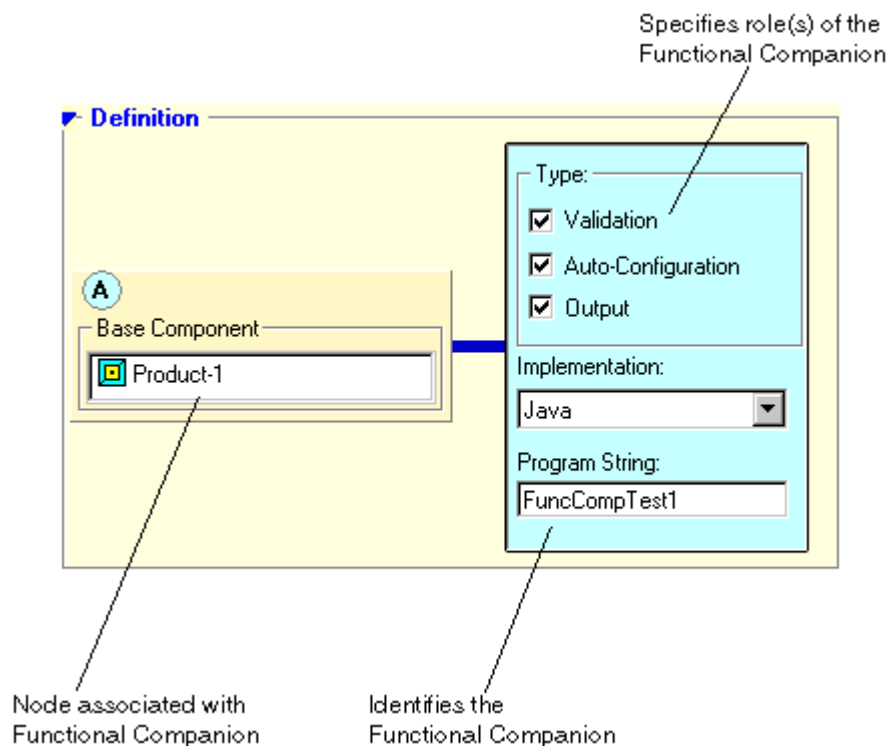
Figure 1-1 *Associating a Component with a Functional Companion*

Figure 1-2 shows details of the Attributes view of the screen in Figure 1-1.

Figure 1–2 Functional Companion Rule: Detail of the Attributes view



1.4.2 Testing Functional Companions in the Oracle SellingPoint Application

Note: The Oracle SellingPoint application is provided with Release 11*i* of Oracle Configurator.

After you generate the Active Model and UI, you can test your Functional Companions by running the Oracle SellingPoint application. You can run the Oracle SellingPoint application in the ways described below.

1.4.2.1 Testing from the Windows Start Menu

To run the Oracle SellingPoint application from the Windows Start Menu:

- Follow the instructions in the *Oracle Configurator and SellingPoint ReadMe* on running the Oracle SellingPoint application.

1.4.2.2 Testing from Oracle Configurator Developer

To run the Oracle SellingPoint application from Oracle Configurator Developer:

1. Use Tools > Options to chose the Oracle SellingPoint application as your test environment.
2. Click the Test button.
3. Follow the instructions in the *Oracle Configurator and SellingPoint ReadMe* on running the Oracle SellingPoint application.

1.4.2.3 Test Functionality in the Oracle SellingPoint Application

The Active User Interface for the Oracle SellingPoint application allows you to test your Functional Companions as appropriate:

Type	User Interface feature
Auto-configuration	A button allows the user to run the <code>autoConfigure()</code> method on the associated Component instance.
Validation	The <code>validate()</code> method is called automatically when the user selects anything.
Output	A button allows the user to run the <code>generateOutput()</code> method on the associated Component instance.

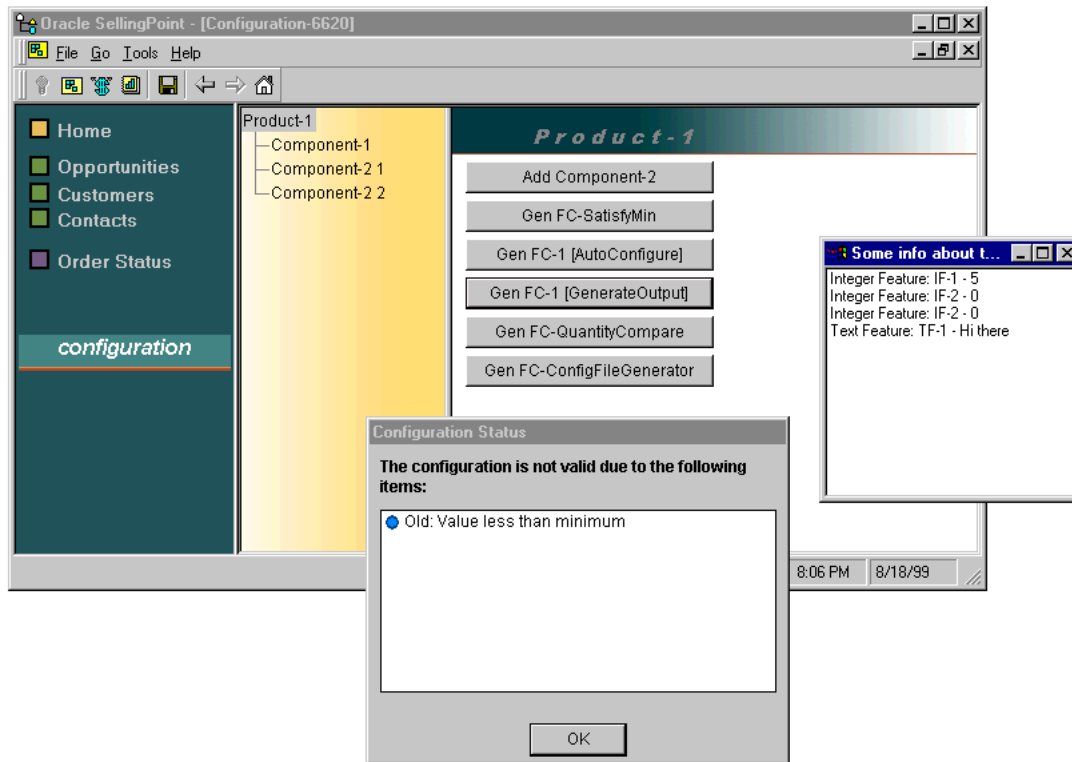
[Figure 1–3](#) illustrates testing several Functional Companions in the Oracle SellingPoint application. The Functional Companions illustrated are the ones defined in the example in [Section 4.2, "Basic Java Functional Companion"](#) on page 4-2.

- Clicking the GenerateOutput Functional Companion button produces a window that displays the current value of several Features. (This uses the "thick client" version of `generateOutput()`. For a thin-client example, see [Section 4.3, "Thin-Client generateOutput\(\) Functional Companion"](#).)
- There is no button for the Validate Functional Companion. The `validate()` method is run whenever there is a change in the value of an Option. If the value

violates a specified range, or a Configuration Rule, then the application displays a Configuration Status message.

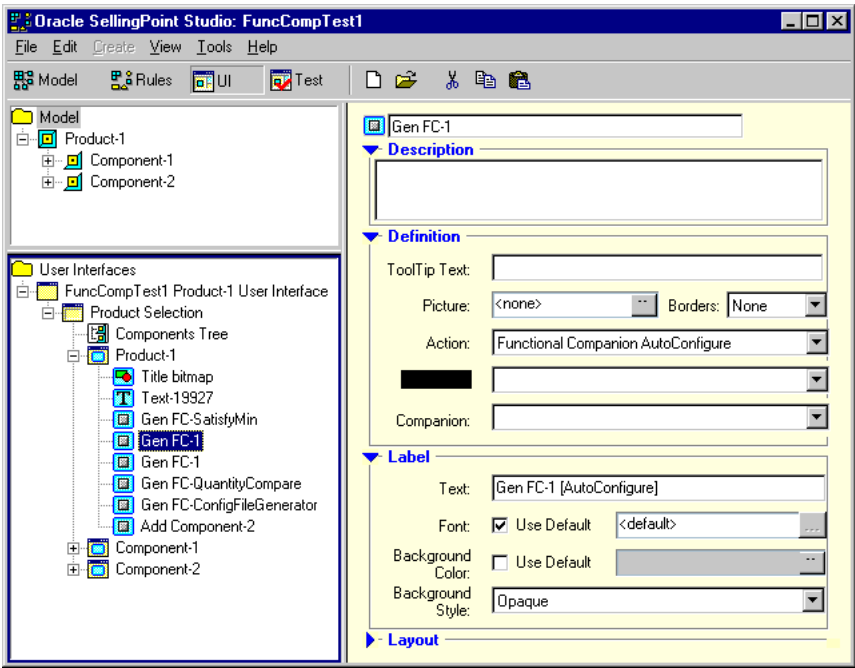
- Clicking the AutoConfigure Functional Companion button changes the value of a numeric Feature (not shown here), in this case violating a specified minimum and thereby triggering the Configuration Status message.

Figure 1–3 Testing Functional Companions in the Oracle SellingPoint application.



Each button that runs a Functional Companion is labelled with default text that identifies the Functional Companion that the button activates. You can use the User Interface module of Oracle Configurator Developer to modify these labels. The labels buttons generated by the Functional Companion shown in [Figure 1–1](#) have been so modified, by adding the self-identifying text [AutoConfigure] and [GenerateOutput], as shown in [Figure 1–4](#).

Figure 1-4 Modifying Functional Companion Buttons



The Configuration Interface Object (CIO)

2.1 Background

2.1.1 What is the CIO?

The Configuration Interface Object (CIO) is an API (application programming interface) that provides your programs access to the Model used by a Oracle SellingPoint application, which you construct with Oracle Configurator Developer.

The CIO is also used by Functional Companions. See [Section 1.2, "Functional Companions and the CIO"](#).

The CIO is a top-level configuration server. The CIO is responsible for creating, saving and destroying objects representing configurations, which themselves contain objects representing Products, Components, Features, Options, Totals and Resources. The runtime configuration model can be completely controlled and manipulated through these interfaces, using methods for getting and setting logical, numeric and string values, and creating optional subcomponents.

Internally, the CIO performs its tasks through interfaces to *logic net objects* (to get and set logic states), to *runtime model subschema objects* (to create the appropriate runtime Model based on the design-time model), and to *configuration subschema objects* (to save and restore configurations created by a user).

The Oracle Configuration Interface Object is written in Java, and implemented as a set of Java packages. The only one that you will usually need to import is:

```
oracle.apps.cz.cio
```

Note: All references in this document to classes, methods, and properties refer to the package `oracle.apps.cz.cio`, and all code examples are in Java, unless otherwise stated.

2.1.2 The CIO and Functional Companions

A Functional Companion is a Java client of the CIO.

Functional Companions are invoked by the CIO through the Oracle SellingPoint application, and Functional Companions call the CIO to get information from the running Model. The CIO is like a broker for the Active Model, in that it passes information both ways. Programmers writing Functional Companions need to have some knowledge of how to use the CIO.

Each Functional Companion is an object class. For every Component instance in your Model that is associated with a Functional Companion, the CIO creates an instance of this class.

2.2 The CIO's Runtime Node Interface Classes

When you program against the CIO, you only create instances of the classes `CIO` (see [Section 2.3, "Initializing the CIO"](#)) and `Configuration` (see [Section 2.4.1, "Creating and Deleting Configurations"](#)). You then use the public interfaces listed in [Table 2-1](#) to access fields in the runtime node objects created by your instances of `CIO` and `Configuration`. Apart from `CIO` and `Configuration`, your code should refer only to these public runtime node interface objects. You should not implement any of the runtime node interface classes, but only use them as references to runtime node objects.

These interfaces are all defined in the Java package `oracle.apps.cz.cio`.

Table 2-1 Runtime node interface classes for the CIO

Interface	Role of implementing classes
IAtp	Implemented by objects that can have ATP calculated.
IBomItem	Implemented by all selectable BOM items.
ICompSetEventListener	Implemented by objects that want to find out about added components.

Table 2–1 (Cont.) Runtime node interface classes for the CIO

Interface	Role of implementing classes
IConfigEventListener	Implemented by objects that want to find out about added components. This listener's methods are called as the result of user interaction, after a functional companion is initialized.
ICount	Implemented by objects that have an associated integer count.
IDecimal	Implemented by objects that have a decimal value.
IDecimalMinMax	Implemented by objects that have a decimal minimum and maximum value.
IFunctionalCompanion	Implemented by Functional Companion objects attached to Components in order to provide programmatic functionality to a configuration model.
IInteger	Implemented by objects that have an integer value.
IIntegerMinMax	Implemented by objects that have an integer minimum and maximum.
IOption	Implemented by objects that act as options. The defining characteristic of an option is that it can be selected and deselected.
IOptionFeature	Implemented by objects that contain selectable options. This interface provides a mechanism for selecting and deselecting options, and for determining which options are currently selected.
IPrice	Implemented by objects that can be priced.
IReadOnlyDecimal	Implemented by objects that have a decimal value.
IRuntimeNode	This interface implements behavior common to all nodes in the runtime configuration tree, including Components, Features, Options, Totals, and Resources.
IState	Implemented by objects that have logic state. This interface contains a set of input states, used to specify a new state for an object, a set of output states, returned when querying an object for its state, and a set of methods for getting and setting the object's state.
IText	Implemented by objects that have a textual value.

The functionality underlying the CIO interfaces is implemented by other classes in `oracle.apps.cz.cio`, which are subject to revision by Oracle. This

interface/implementer architecture protects your code from the effects of such revisions, since the interfaces will remain constant.

Reference

For reference documentation, see: [Package oracle.apps.cz.cio](#).

2.3 Initializing the CIO

In order to use any of the features of the CIO, an application must initialize it, using a JDBC driver to make a connection to the Oracle Configurator schema. This connection enables the CIO to obtain and store data about Model structure, Configuration Rules, and User Interface.

If you are using the CIO in a custom user interface, you will have to initialize the CIO.

Note: When you run Functional Companions through the Oracle SellingPoint application (or test them by using the Test button in Oracle Configurator Developer), this initialization and connection work is automatically handled for you by the application; you do not have to write your own code to initialize the CIO.

1. Import the necessary packages.

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
  
import oracle.apps.cz.cio.*;  
import oracle.apps.cz.common.*;
```

2. Load the database driver that you have installed. For instance, load one of the following:

```
Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");  
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
Class.forName("oracle.jdbc.driver.OracleDriver");
```

3. Create a CZContext context object and pass to it the information needed to make a database connection: the database URL, the user ID and password of the current user, and the owner of the database. The context object manages the

database connection; you should *not* create a separate connection object (e.g., with `java.sql.DriverManager.getConnection`).

```
CZContext contextObject = new CZContext("jdbc:subprotocol:datasource",
    "userID", "password", "schemaOwner");
```

4. Create a CIO object.

```
CIO cioObject = new CIO();
```

[Example 2–1](#) shows how Steps 1 through 4 are combined together. See [Section 4.1, "Initializing the CIO"](#) for a fuller example of initializing the CIO.

Example 2–1 Initializing the CIO (Short Example)

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import oracle.apps.cz.cio.*;
import oracle.apps.cz.common.*;

public class InitCIO
{
    private void InitializeCIO() throws SQLException
    {
        CIO cio;
        CZContext context;

        try{Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");}
        catch (ClassNotFoundException c){System.out.println(c);}
        context = new CZContext("jdbc:oracle:thin:@server01:1521:sid01", "applsypub",
            "pub", "apps");
        cio = new CIO();
    }
}
```

2.4 Access to Configurations

The Configuration object, `oracle.apps.cz.cio.Configuration`, represents a complete configuration. You can use the CIO to work with multiple configurations within the same session.

A configuration communicates through the Configuration object. It supports accessing the containing CIO, the root Component, the project ID, a collection of

current validation failures, access to any runtime node based on its runtime ID, and an indication if the complete configuration is satisfied. In addition, there are methods for starting, ending, and rolling back configuration-level logic transactions; these transactions are to maintain logic consistency and are not database transactions. See [Section 2.4.4, "Logic Transactions"](#).

Reference

For reference documentation, see: [Configuration](#).

2.4.1 Creating and Deleting Configurations

Use `CIO.createConfiguration()` to create a Configuration object, which is the top-level entry point to a configuration. There are different ways to create a Configuration, depending on your requirements.

- To create a Configuration using the name of a Model as specified in Oracle Configurator Developer, use this form:

```
createConfiguration(java.lang.String projectName,  
                   oracle.apps.fnd.common.Context ctx)
```

- To create a Configuration using the ID of the root node of your Model, use this form:

```
createConfiguration(int rootNodeID,  
                   oracle.apps.fnd.common.Context ctx)
```

To determine the root node ID, you would query the Oracle Configurator schema, which is described in the *Oracle Configurator Technical Reference Manual*. Such a query might be:

```
SELECT PS_NODE_ID FROM CZ_PS_NODES WHERE NAME = 'CN9744';
```

Both ways of creating a Configuration object require a database context object, as discussed in Step 3 of [Section 2.3](#) on page 2-4.

For reference documentation on the database context object, see: [CZContext](#).

To delete all runtime structure and memory associated with a configuration, use `CIO.closeConfiguration()`

To get the CIO that created the configuration, use `Configuration.getCIO()`.

Example 2-2 Creating New Configuration Objects

```
// create the database context object
```

```
ctx = new CZContext("jdbc:oracle:thin:@server01:1521:sid01", "aplsyspub",  
"pub", "apps");  
// create Configuration using Project name and Context  
cfg_prj = createConfiguration("Project 10", ctx);  
  
// create Configuration using ID of root node of Model and Context  
cfg_id = createConfiguration(1221, ctx);
```

2.4.2 Saving and Restoring Configurations

Use `Configuration.saveNew()` to Save an entirely new Configuration object into the Oracle Configurator schema.

Use `Configuration.save()` to save subsequent changes to a Configuration object created with `saveNew()`, or to a Configuration object restored with `CIO.restoreConfiguration()`.

Use `CIO.restoreConfiguration()` to restore a Configuration object from the Oracle Configurator schema.

Use `Configuration.saveNewRev()` to save a new revision of the restored Configuration object.

Use `Configuration.saveAs(configHeaderID, revNumber)` to save the current Configuration object over a different Configuration already saved in the database. You use the `configHeaderID` and `revNumber` to open a configuration header object and replace the configuration in it.

Note: Do not save a Configuration object during a logic transaction (see [Section 2.4.4](#)). You may miss some validation messages that are not available until the transaction is committed.

2.4.3 Access to Configuration Parameters

If you are using Oracle Configurator in a web deployment, you can use the CIO to allow a Functional Companion to obtain a list of the configuration inputs (initialization parameters) that were passed from your application to your configuration Model.

Create a Functional Companion that calls `Configuration.getInitParameters()`, which returns a `NameValuePairSet` object. This object contains all the parameter values stored by the Oracle

Configurator UI Server when it processed the initialization message sent by your application to the Oracle Configurator UI Servlet.

See the *Oracle Configurator Custom Web Deployment Guide* for more information.

Reference

For reference documentation, see: [NameValuePairSet](#).

2.4.4 Logic Transactions

In order to help you maintain consistency in interactions with the Oracle Configurator logic engine, you can use *configuration-level logic transactions*. A logic transaction comprises all the logical assertions that constitute a user interaction. At the end of a transaction, the CIO returns a list of all validation failures. See [Section 2.8, "Validating Configurations"](#).

The Configuration object, `oracle.apps.cz.cio.Configuration`, provides a set of methods for starting, ending, and rolling back configuration-level logic transactions. Note that logic transactions are not database transactions.

Inside a transaction, the normal course of action is to set the logical states and numeric values of runtime nodes (see [Section 2.5.5](#) and [Section 2.5.6](#)).

- Use `Configuration.beginConfigTransaction()` to create a new transaction, returning a `ConfigTransaction` object. After performing the desired series of operations (e.g., setting states and values), you must end, commit, or roll back the transaction by passing the `ConfigTransaction` object to one of the mutually exclusive methods that finish the transaction:

```
endConfigTransaction  
commitConfigTransaction  
rollbackConfigTransaction
```

- `Configuration.endConfigTransaction(transaction)` ends the transaction begun with `beginConfigTransaction()`, without committing it (thus skipping validation checking).
- `Configuration.commitConfigTransaction(transaction)` commits the given transaction or series of nested transactions, propagates the effect of user selections throughout the configuration Model, and triggers validation checking (see [Section 2.8, "Validating Configurations"](#)).
- `Configuration.rollbackConfigTransaction(transaction)` rolls back the unfinished transaction, undoing the operations performed inside it.

You can nest intermediate transactions with `beginConfigTransaction()` and `endConfigTransaction`, delaying validation checking until you call `commitConfigTransaction()`. You must end or commit inner transactions before ending or committing the outer ones that contain them. When rolling back unfinished transactions, with `rollbackConfigTransaction()`, you can roll back outer transactions, which automatically rolls back the inner transactions.

When beginning a transaction, you can autocommith it, by setting the optional boolean `autoCommit` argument to `beginConfigTransaction()` to `TRUE`. If no argument is set, then the transaction inherits the autocommit state of its parent (outer) transaction. If an outer transaction sets `autoCommit` to `TRUE`, then inner transactions can override it to either `TRUE` or `FALSE`. If an outer transaction sets `autoCommit` to `FALSE`, then inner transactions cannot override it; they will always inherit `FALSE`.

Reference

For reference documentation, see: [Configuration](#).

2.5 Access to Nodes of the Model at Runtime

The root Component, and every other node in the underlying runtime Model tree, implements the `IRuntimeNode` interface. This interface exposes the type of the node (based on a set of node type constants), its name, the database ID, the database node of which this runtime node is an instance, a runtime ID that is unique to this node across all nodes created by this particular CIO, the parent node (which is null for the root Component), a (possibly empty) collection of children, and information about whether this part of the runtime tree has been satisfied. See [Section 2.6, "Introspection through IRuntimeNode"](#).

Use `IRuntimeNode.getConfiguration()` to get the configuration to which a node belongs.

Reference

For reference documentation, see: [IRuntimeNode.getConfiguration\(\)](#).

2.5.1 Opportunities for Modifying the Model

There are some restrictions on when you can modify the Model.

- Normally, you should use an Auto-configuration Functional Companion to perform such modifications. To do so, you perform the actions in the body of the `IFunctionalCompanion.autoConfigure()` method. Your end user

invokes the Auto-configuration companion by clicking a button in the User Interface. See ["The `autoConfigure\(\)` Interface Method"](#) on page 2-25.

- If you want to modify the Model automatically when a configuration instance is created, restored, or saved (for instance, to perform certain selections when the configuration session begins or ends), you must use the methods of the class `AutoFunctionalCompanion`. For reference, see [AutoFunctionalCompanion](#) on page B-5.
- You should never modify the Model in a Initialization Functional Companion (i.e., in the `IFunctionalCompanion.initialize()` method). See ["The `initialize\(\)` Interface Method"](#) on page 2-23.
- You should never modify the Model in a Validation Functional Companion (i.e., in the `IFunctionalCompanion.validate()` method). See ["The `validate\(\)` Interface Method"](#) on page 2-26.

2.5.2 Accessing Components

Use `Component.getFunctionalCompanions()` to return a list of all the Functional Companions associated with this Component.

2.5.3 Adding and Deleting Optional Components

The `ComponentSet` represents a set of similar Components that can be added and deleted dynamically. Each `ComponentSet` implements the `IRuntimeNode` interface.

Use `ComponentSet.add()` to add an optional Component. The `add()` method can throw a `LogicalException` exception if adding the Component causes a logical contradiction.

Use `ComponentSet.delete(component)` to delete an optional Component.

2.5.4 Accessing Features

There are several specialized types of Features. Each Feature type supports the `IRuntimeNode` interface, enabling you to use its general methods for working with runtime nodes (see ["Introspection through `IRuntimeNode`"](#) on page 2-15). Each type also supports its own interface with appropriately specialized methods.

`BooleanFeatures` have a boolean (true/false) value.

`CountFeatures` have both a boolean value, and an associated integer-valued numeric count. The minimum value of the count must be greater than or equal to

zero. The boolean value of a `CountFeatures` object is returned by its methods `hasMax()` and `hasMin()`.

`IntegerFeatures` have an integer numeric value. The value can be positive, negative, or zero.

`DecimalFeatures` have a floating point value.

`TextFeatures` have a string value.

`OptionFeatures` have a logic value, and a set of options as children. You can use the methods `getMinSelected()` and `getMaxSelected()`, of `IOptionFeature`, to determine the minimum and maximum number of a Feature's child Options that can be selected. If you do, first use `hasMinSelected()` or `hasMaxSelected()` to determine whether there is a minimum or maximum number of Options.

Note: If, in Oracle Configurator Developer, you set the minimum count of a Feature greater than or equal to zero, then the CIO treats this as a `CountFeature` object. If you set the minimum count less than zero, then the CIO treats this as a `IntegerFeature` object.

2.5.5 Getting and Setting Logic States

- To interact with objects that have logic state, you implement the `IState` interface. This interface contains:
- a set of input states, used to specify a new state for an object
- | | |
|--------|--|
| FALSE | The input state used to set an object to false. |
| TRUE | The input state used to set an object to true. |
| TOGGLE | The input state used to turn an object state to true if it is false or unknown, and to make it unknown or false if it is true. |
- a set of output states, returned when querying an object for its state
- | | |
|--------|--|
| LFALSE | The logically false output state, indicating that the state is false as a consequence of a rule. |
| LTRUE | The logically true output state, indicating that the state is true as a consequence of a rule. |
| UFALSE | The user false output state, indicating that a user has set this object to false. |

UNKNOWN	The unknown output state.
UTRUE	The user true output state, indicating that a user has set this object to true.

- a set of methods for getting and setting the object's state

getState()	Gets the current logic state of this object.
setState(int state)	Change the current logic state of this object.

The code fragment in [Example 2-3](#), which uses `getState()` with `UTRUE`, is taken from [Section 4.2, "Basic Java Functional Companion"](#), after the comment `"/get the necessary components from the configuration // line 61"`.

Example 2-3 Getting the state of a node

```
//get the necessary components from the configuration    // line 61
baseComponent = (oracle.apps.cz.cio.Component)comp_
node.getChildByName("Component-1");
    of = (OptionFeature)baseComponent.getChildByName("Feature-1");
    op = (Option)of.getChildByName("Option-1");
    intFeat = (IntegerFeature)baseComponent.getChildByName("IF-1");
    //check if the option is set to UTRUE. If so, set the Integer value
to 5
    if( op.getState() == IState.UTRUE )
        intFeat.setIntValue(5);
    }
```

Example 2-4 Setting the state of a node

The following code fragment, which uses `setState()` with `TOGGLE`, toggles the state of the selected item in the Model tree.

```
private void toggleSelectedItem() {
    IState node = (IState)tree.getLastSelectedPathComponent();
    try {
        node.setState(IState.TOGGLE);
    }
    catch (LogicalException le) {}
    catch (TransactionException te) {}

    tree.repaint();
}
```

2.5.6 Getting and Setting Numeric Values

You can use the following methods to get and set the values of objects that have numeric values.

For decimal values, use:

```
IDecimal.setDecimalValue()  
IReadOnlyDecimal.getDecimalValue()
```

For integer values, use:

```
IInteger.setIntValue()  
IInteger.getIntValue()
```

The code fragment in [Example 2-5](#) uses `setIntValue()` to change the value of an Integer Feature. Note that you can use the generalized `IRuntimeNode` interface for flexibility in selecting a child node, and then cast the node object to a particular interface to perform the desired operation on it.

Example 2-5 Setting a numeric value

```
// select a node by name  
IRuntimeNode limit = baseComp.getChildByName("Current Limit");  
  
// use an interface cast to set the node's value by the desired type  
((IInteger)limit).setIntValue(5);
```

To determine whether a numeric value has violated its Minimum or Maximum range, you may need to iterate through the collection of validation failures returned by `Configuration.getValidationFailures()` after setting a value, for instance with `IInteger.setIntValue()`. See [Section 2.8, "Validating Configurations"](#) for more background.

There is a subtlety that you should take note of.

`IDecimal.setDecimalValue()` does not throw a `LogicalException` when setting the value of a decimal feature that exceeds the feature's Min/Max limits. The collection of validation failures returned by `Configuration.getValidationFailures()` does not include any failures that result from setting a numeric value until the logic transaction has been closed,

so there is no way to roll back a transaction in which a Min/Max violation has occurred. Here is a suggested method for dealing with this situation:

1. Open a transaction.
2. Set the new value.
3. Close the transaction.
4. Get the collection of validation failures for the configuration.
5. If the last transaction caused a Min/Max violation, then call `Configuration.undo()`, which retracts the last transaction.

This situation illustrates why it is a good practice to perform the setting of a single value inside a logic transaction. You can always undo it if the result is unsatisfactory.

2.5.7 Accessing Properties

You can determine which Properties belong to a runtime node, then use methods of the class `Property` to obtain information about the Properties.

Use `IRuntimeNode.getProperties()` to get a collection of the properties associated with a node.

Use `IRuntimeNode.getPropertyByName()` to get a particular property of a node, based on its name.

When you have the `Property`, use methods of the class `Property`, such as `getStringValue()`, to obtain specific information.

2.5.8 Access to Options

Option features have special methods for selecting options and querying for selected options. The `selectOption()` method implements mutual exclusion behavior for option features with a min/max of 1/1 by deselecting a currently selected option before selecting the new option. The `getSelectedOption()` method throws the `TooManySelectedException` if more than one option is selected in the feature.

An option is a child of an option feature which supports a true/false logic state and a count. Options implement the `IRuntimeNode` interface.

You can use the interface class IOption to select, deselect, and determine the selection state of Options.

Table 2–2 Methods of the Interface Class IOption

Method	Action
deselect()	Deselect this Option.
isSelected()	Returns true if this Option is selected, and false otherwise.
select()	Select this Option.

The code fragment in [Example 2–6](#) displays a “check” icon if an Option of a runtime node is selected, and displays an “unsatisfied” icon if the node is logically unsatisfied:

Example 2–6 Testing whether an option is selected, or satisfied

```
IRuntimeNode rtNode = (IRuntimeNode)value;
if (value instanceof IOption) {
    IOption optionNode = (IOption)value;
    if (optionNode.isSelected()) {
        setIcon(checkIcon);
    }
} else if (rtNode.isUnsatisfied()) {
    setIcon(unsatIcon);
}
return this;
```

2.6 Introspection through IRuntimeNode

You can get information about a node in a Model at runtime by using methods of the interface class IRuntimeNode. This helps you to write “generic” Functional Companions, which can interact with a Model tree dynamically, without having prior knowledge of its structure.

Table 2–3 Methods of the interface class IRuntimeNode

Method	Action
getChildByID(id)	Gets a particular child identified by its ID.
getChildByName(name)	Gets a particular child identified by its name.
getChildren()	Gets the children of this runtime configuration node.

Table 2–3 (Cont.) Methods of the interface class IRuntimeNode

Method	Action
getChildrenByType(type)	Gets all of the children of a particular type.
getConfiguration()	Gets the configuration to which this node belongs.
getDatabaseID()	Gets the database ID of the node. This is the field CZ_PS_NODES.PS_NODE_ID in the Oracle Configurator schema, described in the <i>Oracle Configurator Technical Reference Manual</i> .
getDescription()	Returns the design-time description of the runtime node.
getName()	Gets the name of the node.
getParent()	Gets the parent of the node.
getProperties()	Returns a collection of the properties associated with this node. The collection contains items of the type Property.
getPropertyByName(name)	Returns a particular property of the node, based on its name. Returns null if a property of the given name does not exist.
getRuntimeID()	Gets the runtime ID of the node.
getSelectionLineID()	Returns selection line ID (configuration output database ID) for node.
getType()	Gets the type of this node.
hasCount()	Returns true if the node has an object count.
hasDecimalValue()	Returns true if the node has a decimal value.
hasDescription()	Returns true if there is a design-time description of the runtime node.
hasSelectionLineID()	Returns true if node has a selection line ID (configuration output ID), false if no
hasState()	Returns true if the node has a logical state.
hasTextValue()	Returns true if the node has a text value
isNative()	Returns true if this is a native BOM node.
isUnsatisfied()	Returns true if this particular node, or any one of its children, has not been completely configured.
isUnsatisfiedNode()	Returns true if this particular node has not been completely configured.

Table 2–3 (Cont.) Methods of the interface class IRuntimeNode

Method	Action
toString(description)	Returns a String representation of this node, based on whether the client demands a description (if there is one) or just a name.

Reference

For reference documentation, see the Methods summary for: [IRuntimeNode](#).

The code fragment in [Example 2–7](#) displays a “check” icon if an Option of a runtime node is selected, and displays an “unsatisfied” icon if the node is logically unsatisfied:

Example 2–7 Testing whether a node is selected, or satisfied

```
IRuntimeNode rtNode = (IRuntimeNode)value;
    if (value instanceof IOption) {
        IOption optionNode = (IOption)value;
        if (optionNode.isSelected()) {
            setIcon(checkIcon);
        }
    } else if (rtNode.isUnsatisfied()) {
        setIcon(unsatIcon);
    }
    return this;
```

The code fragment in [Example 2–8](#) creates a Configuration object config, sets homeTheater to the root Component of the configuration, and sets userType to the child node with the user-visible name “User Type”.

Example 2–8 Getting a child node by name

```
Configuration config = m_cio.createConfiguration(m_product);
IRuntimeNode homeTheater = config.getRootComponent();

IRuntimeNode userType = homeTheater.getChildByName("User Type");
```

The code fragment in [Example 2–9](#), which uses `getChildrenByType()`, is taken from [Section 4.2, “Basic Java Functional Companion”](#), after the comment “//get all the text features // line 167”.

Example 2–9 Collecting all child nodes by type

```
//get all the text features // line 167
    textFeatList = comp.getChildrenByType(comp.TEXT_FEATURE);
    traverseTree(comp.getChildren(),
                comp.TEXT_FEATURE,
                textFeatList);
    iter = textFeatList.iterator();
```

2.7 Handling Logical Contradictions

When you make a request to modify the state of a logic network, for instance by using `ISate.setState(state)`, the result may be a failure of the request because of a logical contradiction. Such a failure will create and throw a *logical exception*, accessed through either the `LogicalException` or `LogicalOverridableException` objects. A `LogicalException` cannot be overridden.

See "[Overriding Contradictions](#)" for details on using `LogicalOverridableException` to override the contradiction.

Use `LogicalException.isOverridable()` to determine whether the exception is an instance of `LogicalOverridableException`, which can be overridden with its `override()` method.

Use `LogicalException.getCause()` to get the runtime node that caused the failure.

Use `LogicalException.getReasons()` to get a list of reason strings for the failure.

Use `LogicalException.getMessage()` to provide a message containing either the cause or the reasons.

Reference

For reference documentation, see: [LogicalException](#).

2.7.1 Generating Error Messages from Contradictions

You can use the `Reason` object to wrap the information returned by a contradiction, in order to include information about internal error messages.

```
Reason(int type,
      IRuntimeNode node,
```

```
java.lang.String msg)
```

Constructs a Reason given all of its information:

type What type of reason this is.

node The node that caused the problem.

msg The message returned.

Use `Reason.getMessage()` to get the message associated with this reason.

Use `Reason.getNode()` to get the node associated with this reason.

Use `Reason.getType()` to get the type of reason held in this object.

Use `Reason.toString()` to convert this object to a string.

Reference

For reference documentation, see: [Reason](#).

2.7.2 Raising Exceptions

When a Functional Companion is invoked, the Oracle Configurator UI Server wraps a transaction around this invocation. A contradiction that is not handled in your code, and is not rolled back inside the Functional Companion, may result in a fatal exception. If there is a fatal exception, the UI Server kills the configuration session (dropping any open transactions), and displays a message to the end user, who cannot proceed with the configuration.

In the current version of the CIO, you can raise a `FuncCompMessageException`, which allows you to present a dialog box displaying a specified message, and the name of the Functional Companion that raised the exception. When the end user dismisses the dialog box, the UI Server commits the open CIO transaction, and allows the end user to proceed with the configuration. It is possible that the Model can be left in an uncertain state.

This situation can be a particular issue for auto-configuration Functional Companions.

Reference

For reference documentation, see: [FuncCompMessageException](#).

2.7.3 Overriding Contradictions

Your Oracle SellingPoint application or Functional Companion can provide a message to your user, and ask whether the contradiction should be overridden.

If a logical contraction can be overridden, then a `LogicalOverridableException` is signalled, instead of a `LogicalException`.

`LogicalOverridableException` is a subclass of `LogicalException` that adds an `override()` method. Use

`LogicalOverridableException.override()` to override the contradiction.

Both types of exceptions (`LogicalException` and `LogicalOverridableException`) may be thrown back from any of the "set" methods (like `setState()`) or from

`Configuration.commitConfigTransaction(transaction)`. If you want to override the overridable exception you have to call its `override()` method, which can also throw a `LogicalException`. This means that even when you try to override the exception you still trigger a contradiction and cannot continue. If the override succeeds then you still need to call `commitConfigTransaction()` to close the transaction. If you don't want to override or if you get a `LogicalException` you need to call `rollbackConfigTransaction()` to purge it. [Example 2-10](#) is a code fragment that illustrates this point. Note that the operations represented with `<ASK "text">` and `<SHOW "text">` are not part of the CIO but suggest where your own Functional Companion should try to handle the situation.

Example 2-10 Handling and overriding Logical Exceptions

```
try {
    // begin a transaction
    ConfigTransaction tr = config.beginConfigTransaction();

    // call the "set" method
    opt1.setState();

    // commit the transaction
    config.commitConfigTransaction(tr);
}
catch(LogicalOverridableException loe) {
    proceed = <ASK "Do you want to override?">;
    if (! proceed) {
        rollbackConfigTransaction();
    }
    else {
```

```

    try {
        // override the contradiction and ...
        loe.override();
        // ... finish the transaction
        commitConfigTransaction();
    }
    catch (LogicalException le) {
        // we cannot do anything
        <SHOW "Cannot be overridden">
        config.rollbackConfigTransaction(tr);
    }
}
}
catch (LogicalException le) {
    // we cannot do anything
    <SHOW "Cannot be overridden">
    config.rollbackConfigTransaction(tr);
}

```

Reference

For reference documentation, see: [LogicalOverridableException](#).

2.8 Validating Configurations

You want to be able to check whether a Configuration is valid (that is, does not violate the rules associated with it).

The CIO validates a Configuration after all logical assertions that constitute a user interaction are performed. This corresponds exactly to the length of a logical transaction. See [Section 2.4.4, "Logic Transactions"](#).

Validation checking and reporting occur when a logical transaction is ended by using `Configuration.commitConfigTransaction(transaction)` or `Configuration.rollbackConfigTransaction(transaction)`.

After a committal or rollback, the CIO traverses the nodes of the Model, checking for validation failures, selected items and unsatisfied items. These are kept in a set of collections maintained on the Configuration.

At this point, you can call the following methods of `oracle.apps.cz.cio.Configuration`:

<code>getValidationFailures()</code>	Actually returns a collection of "ValidationFailure" objects. Call this after committing or rolling back a transaction, in order to inspect the list of validation failures.
<code>getSelectedItems()</code>	Returns a collection of selected items as a <code>StatusInfo</code> structure indicating the set of selected (true) items in the Configuration.
<code>getUnsatisfiedItems()</code>	Returns a collection of unsatisfied items as a <code>StatusInfo</code> structure indicating the set of unsatisfied items in the Configuration.

As nodes become selected they have a status of `STATUS_NEW`. If they continue to be selected since the last transaction their status is `STATUS_EXISTING`. If they become unselected, their status becomes `STATUS_DELETED` until the next transaction at which time they will be removed from the collection.

If you are writing a Functional Companion, the `validate()` method should return a list of `CompanionValidationFailure` objects in the event of a validation failure. This allows you to return more than one failure. Your `validate()` method can include several tests; you can track which ones failed, and why. See [Section 2.9.3, "The validate\(\) Interface Method"](#).

Reference

For reference documentation, see: [ValidationFailure](#) and [Configuration](#).

2.9 Standard Interface Methods for Functional Companions

You provide functionality for your Functional Companion by implementing body code for the methods described in this section. For particulars that apply to the languages currently supported by the CIO, and examples, see [Section 1.3, "Building Functional Companions in Java"](#).

These methods are invoked by your Oracle SellingPoint application, through the CIO, in response to program events or the actions of end users. The type of method invoked for each Component is determined when you associate the Component with a Functional Companion in Oracle Configurator Developer. See [Section 1.4, "Incorporating Functional Companions in your Application"](#) for details.

These methods are invoked by the CIO for each Functional Companion object that it creates for the Components in your Model. Note that your code does not invoke

these methods directly; that is done by the CIO. Rather, you implement the body of each method, using the API provided by the CIO to communicate with your Model.

The body of any or all of these methods can be empty. Your Functional Companion object has to implement only those methods indicated in Oracle Configurator Developer.

The interface class that defines these methods is:

`oracle.apps.cz.cio.IFunctionalCompanion`

Reference

For reference documentation, see: [IFunctionalCompanion](#).

Table 2–4 *Standard methods of the IFunctionalCompanion interface*

Method	Purpose	Details in
<code>initialize</code>	Saves information about the Model and performs any actions needed to initialize the Functional Companion.	Section 2.9.1
<code>autoConfigure</code>	Performs a programmatic configuration step.	Section 2.9.2
<code>validate</code>	Programmatically checks that a configuration is valid and throws a <code>LogicalException</code> object if the Model is not valid.	Section 2.9.3
<code>generateOutput</code>	Generates output for this Component, for either a thick or thin client.	Section 2.9.4
<code>terminate</code>	Performs any cleanup on this Functional Companion that needs to occur before the Companion is destroyed.	Section 2.9.5

2.9.1 The `initialize()` Interface Method

The `IFunctionalCompanion.initialize()` method is called when the companion is created. It connects a Functional Companion object to its configuration modeling environment (for example, a running instance of the Oracle SellingPoint application). Be aware that Functional Companions are created and initialized after all subcomponent instances are created for the current Component instance.

Your implementation of `initialize()` can include tasks that you wish to perform when the Functional Companion is first created. For example, you might wish to start writing audit messages to a log file, tracking the actions performed by your end users.

When an Oracle SellingPoint application runs, it creates runtime instances of all the Components in the Model and their associated Functional Companions. When a Functional Companion object is created, the CIO calls `initialize()` and passes the following input parameters:

Name	Type	Description
node	IRuntimeNode	The node instance associated with the Functional Companion being created. Specified in Configurator Developer. Currently, only Components can be specified in Configurator Developer.
name	String	The name of the Functional Companion. Specified in Configurator Developer.
description	String	A description of the Functional Companion. Specified in Configurator Developer.
id	int	The database ID of the Functional Companion. Created internally.

Note: It is worth emphasizing that the node passed as the first input parameter to `initialize()` is specified in Oracle Configurator Developer, when you create the Functional Companion rule that associates a Model node with your Functional Companion.

Your Functional Companion should ordinarily never directly call `FunctionalCompanion.initialize()`, since the CIO does that for you automatically. However, if your Functional Companion extends `FunctionalCompanion` as its base class, and you wish to perform some specialized initialization tasks, then the overriding `initialize()` method in *your* class should call `super.initialize()`. This passes some necessary variables to the superclass (`oracle.apps.cz.cio.FunctionalCompanion`) so that its methods will work.

It is not normally necessary to implement your own `initialize()` method in your Functional Companion. If you need to obtain the values of the input parameters of `FunctionalCompanion.initialize()` for use elsewhere in your Functional Companion, you can use the set of accessor methods of `FunctionalCompanion` already provided in the `oracle.apps.cz.cio.FunctionalCompanion` base class. Each of these methods returns the value of the corresponding input parameter:

<code>getRuntimeNode()</code>	Returns the runtime node to which this functional is associated.
<code>getName()</code>	Returns the name of the functional companion.
<code>getDescription()</code>	Returns the description of the functional companion.
<code>getID()</code>	Returns the database ID of the functional companion.

Note: Currently, in Configurator Developer, you can only associate a Functional Companion with a Component (which corresponds to the `node` parameter of `initialize()`). However, to accommodate possible future enhancement of Configurator Developer, the `IFunctionalCompanion` interface allows any runtime node to be associated with your Functional Companion.

Reference

For reference documentation, see: `initialize(IRuntimeNode, String, String, int)`.

2.9.2 The `autoConfigure()` Interface Method

The `IFunctionalCompanion.autoConfigure()` method is called at the request of the controlling User Interface, and can set states in the Model, add optional Components, and other tasks that modify the Model.

This method performs an automatic configuration on the Model. This action can include changing the logical state of Options, or adding nodes underneath the selected Component instance in the Model tree.

Your implementation of `autoConfigure()` can include configuration actions that you wish to be performed before your end users arrive at a certain point in a configuration session, or as the result of certain choices that they make.

Note: In the current version of Release 11*i*, a contradiction that is not handled in your code, and is not rolled back, may result in a fatal exception, since the Oracle Configurator UI Server may not be able to commit the overarching transaction, or provide adequate feedback. This can be an issue for auto-configuration Functional Companions.

For reference documentation, see:
[FuncCompMessageException](#).

Warning: Ordinarily, you cannot use an external program to modify an open configuration asynchronously. Doing so circumvents the Oracle Configurator UI Server, which can result in problems with multi-threading and updating of the display in the user interface. The key exception to this restriction is the execution of the body of the `autoConfigure()` method. When you use `autoConfigure()` to modify the Model, you must finish all modifications before returning from the method. This allows the UI Server to update the display to reflect any modifications.

Reference

For reference documentation, see: [autoConfigure\(\)](#).

2.9.3 The `validate()` Interface Method

The `IFunctionalCompanion.validate()` method is called automatically when a logical transaction takes place, and should return a `List` of `CompanionValidationFailure` objects if the Model is not valid.

This method performs a functional validation for the selected Component instance each time the end user selects a node in the Model (for example, in the Configurations section of the Oracle SellingPoint application).

You should not modify the Model in a validation function. Doing so can cause unexpected application failures.

Your implementation of `validate()` can include tasks that you wish to perform whenever your end users make any selection. For example, you might wish to perform a calculation based on the object count of the selected Component, and

present the end user with a notification if the result is outside a range that you define.

If the validation fails, then information about the failure is gathered by the CIO in a `List` of `CompanionValidationFailure` objects.

The general structure of your implementation of `validate()` should be:

1. Collect inputs from the `Model`.
2. Call a generic validation function that you define outside the body of `validate()`.
3. Propagate the result back as the value of the function, either `null` or a `List` of `CompanionValidationFailure` objects.

Reference

For reference documentation, see: [validate\(\)](#).

2.9.4 The `generateOutput()` Interface Method

The `generateOutput()` method is invoked at the request of the controlling `User Interface`.

Your implementation of `generateOutput()` might include tasks such as writing to a database, creating a report, or producing a visualization of the end user's configuration choices.

There are two versions of `generateOutput()`:

- “thick client” version

```
public String generateOutput();
```

A thick client architecture is one in which the configuration `Model`, and the user interface for manipulating it, both reside on the same client machine. The thick client architecture allows your `Functional Companion's Output` method to produce output windows directly on the client machine.

This version is invoked when your `Functional Companion` operates with the `Oracle SellingPoint` application. (Note: The returned string is ignored.)

- “thin client” version

```
public void generateOutput(HttpServletResponse response) throws IOException
```

A thin client, browser-based architecture is one in which the configuration Model resides on a server, and the user interface resides on a client machine's web browser. The thin-client architecture allows your Functional Companion's `Output` method to produce out in web-browser windows.

This version is invoked when your Functional Companion operates in a web deployment.

See [Section 4.3, "Thin-Client `generateOutput\(\)` Functional Companion"](#) for an example.

Currently, there is no mechanism for output generated through `generateOutput()` to provide feedback to the User Interface or the runtime Model.

Reference

For reference documentation, see: [`generateOutput\(\)`](#) and [`generateOutput\(HttpServletResponse\)`](#).

2.9.5 The `terminate()` Interface Method

The `IFunctionalCompanion.terminate()` method is called automatically by the CIO when the Component that the Functional Companion is attached to is deleted from the running Model.

Your implementation of this method can include tasks that you wish to perform when the Functional Companion is deleted. For example, if `initialize()` opens a file and reads some data, `terminate()` would close the file.

Your Functional Companion should ordinarily never directly call `FunctionalCompanion.terminate()`, since the CIO does that for you automatically. However, if your Functional Companion extends `FunctionalCompanion` as its base class, and you wish to perform some specialized termination tasks, then the overriding `terminate()` method in *your* class should call `super.terminate()`.

Reference

For reference documentation, see: [`terminate\(\)`](#).

Reference Documentation for the CIO

Reference documentation for the Oracle Configuration Interface Object is provided in the form of pages generated by the Javadoc tool from the source code for the CIO.

For the main entry point to these pages, follow this link:

- [CIO Package and Related Classes](#)

Tips

Here are some tips on using the generated reference documentation:

- Use the Bookmarks pane to navigate through the reference.
- Use the Contents and Index to look up items alphabetically.
- Reminder: Constants are referred to in Java as “static variables,” and are listed under the heading “Fields” in the class in which they are defined.

This chapter contains code examples illustrating the use of Functional Companions and the CIO. These examples are fuller and longer than the examples provided in the rest of this document, which are often fragments. The examples here can be compiled and used. See the cited background sections for details.

The examples given here are all in Java, and were compiled with JDK 1.1.8.

4.1 Initializing the CIO

For background, see [Section 2.3, "Initializing the CIO"](#). This example is intended for custom user interfaces that use the CIO.

Example 4–1 Initializing the CIO (Long Example)

```
import java.io.*;
import java.sql.*;
import oracle.apps.cz.cio.*;
import oracle.apps.cz.common.*;

class cioExample
{
    private CIO InitializeCIO()
    {
        CIO cio = null;
        CZContext context = null;

        String jdbcDriver      = "oracle.jdbc.driver.OracleDriver";    // Class
name of the JDBC driver
        String dbURL           = "jdbc:oracle:thin:@server01:1521:sid01";
        String dbOwner          = "apps";
    }
}
```

```
String dbUsername      = "applsypub";
String dbPassword      = "pub";

try {
    // Load the JDBC driver
    Class.forName(jdbcDriver);

    // Establish a connection to the database
    context = new CZContext(dbURL, dbUsername, dbPassword, dbOwner);
}
catch (ClassNotFoundException cnfe) {
    System.out.println("Error loading class " + jdbcDriver);
    System.exit(0);
}
catch (SQLException sqle) {
    System.out.println("Error in creating Context");
    System.exit(0);
}

try {
    // Initialize the CIO
    cio = new CIO();
}
catch (Exception e) {
    System.out.println("Exception in InitializeCIO");
    cio = null;
}

return cio;
}
}
```

4.2 Basic Java Functional Companion

For background, see [Section 1.3, "Building Functional Companions in Java"](#).

[Example 4-2](#) implements all of the types of Functional Companions, which are described in [Section 1.1.1](#). The example implements the methods described in [Section 2.9](#), and assumes the structure of the Model shown in [Figure 1-1](#).

Example 4-2 Basic Functional Companion: FuncCompTest1

```
import oracle.apps.cz.cio.*;
import com.sun.java.util.collections.List;           // line 2
import com.sun.java.util.collections.ArrayList;
```

```

import com.sun.java.util.collections.Iterator;
import java.awt.*;
import java.awt.event.*;

public class FuncCompTest1 extends FunctionalCompanion
{
    oracle.apps.cz.cio.IRuntimeNode comp_node;    // currently, only Components

    Frame f;
    java.awt.List uiList;

    /**
     * Constructor:
     * Can be used for any necessary setup.
     */
    public FuncCompTest1()
    {
    }

    /**
     * Initialize: calls 'super' to get access to its functions.
     * @param comp_node - base node of functional companion (currently, only
Components)
     * @param name - the name of the companion
     * @param description - a description of the companion
     * @param id - the db id of the companion
     * All of these parameters, except 'id', are specified in the companion
     * definition in Developer. id is created internally.
     */
    public void initialize(IRuntimeNode comp_node, String name, String
description, int id)
    {
        this.comp_node = comp_node;
        super.initialize(comp_node, name, description, id); // line 35
    }

    /**
     * Functionality implementation:
     * There are three types of functionality for companions
     * Any number of them can be implemented in each companion.
     */

    /**
     * Type 1: Auto-configuration
     * If this method is defined, a button will appear in the UI and the code

```

```
will
    * be run when the button is clicked. It is used to make changes to the
    * configuration.
    */
    public void autoConfigure()
        //This example simply checks if a certain value is true and, if so, sets
        //an integer feature value to 5;
    {
        OptionFeature of;
        Option op;
        IntegerFeature intFeat;
        oracle.apps.cz.cio.Component baseComponent;

        try
        {
            //get the necessary components from the configuration // line 61
            baseComponent = (oracle.apps.cz.cio.Component)comp_
node.getChildByName("Component-1");
            of = (OptionFeature)baseComponent.getChildByName("Feature-1");
            op = (Option)of.getChildByName("Option-1");
            intFeat = (IntegerFeature)baseComponent.getChildByName("IF-1");
            //check if the option is set to UTRUE. If so, set the Integer value
to 5
            if( op.getState() == IState.UTRUE )
                intFeat.setIntValue(5);
        }
        catch(Exception e){System.out.println(e);}
    }

    /**
    * Type 2: Validation
    * If this method is defined, the code will automatically be run any time
    * a change is made to the base node or one of its children in the
    * configuration. It is used for ensuring that changes made result in a
    * valid configuration. If not, the method returns a list of validation
failures.
    */
    public List validate()
        //This example defines 'min' (presumably the minimum amount this
customer
        //may order) and checks to see if the value equals at least this amount.
        //In the real world, you would want to get this value from your customer
        //database. For example, customers in foreign countries may have higher
minimums
        //since shipping is expensive.
```

```

{
    int min = 8;
    int val = 0;
    IntegerFeature intFeat;
    ArrayList failures = new ArrayList();

    try
    {
        //get the value of the integer feature in the configuration
        oracle.apps.cz.cio.Component c = (oracle.apps.cz.cio.Component)comp_
node.getChildByName("Component-1");
        intFeat = (IntegerFeature)c.getChildByName("IF-1");
        val = intFeat.getIntValue();
    }
    catch(NoSuchChildException e){e.printStackTrace();}

    //check to see if the value in the config is not at least the min value
// line 102
    if( !(val >= min) )
        failures.add( new CompanionValidationFailure("Value less than
minimum", comp_node, this) );

    if(failures.isEmpty())
        return null;
    else
        return failures;
}

/**
 * Type 3: Output
 * If this method is defined, a button will appear in the UI which, when
 * pressed, will run the code below. It is used to generate output to the
 * user. Note: this uses the "thick client" version of generateOutput().
 */
public String generateOutput()
    //This example opens up a window with a list of some of the current
    //components of the configuration and their values. This example is
very basic but
    //the idea here is that data from the configuration can be used to
generate reports,
    //graphs, models, etc.
    {
        IntegerFeature intFeat;
        TextFeature textFeat;

```

```
com.sun.java.util.collections.List intFeatList, textFeatList;
Iterator iter;

//setup the UI
if(f == null)
{
    f = new Frame("Some info about this config");
    uiList = new java.awt.List();
    f.add(uiList);
    f.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e)
        {
            f.dispose();
        }
    });
}
uiList.removeAll();

try
{
    //get all the integer features
    intFeatList = comp_node.getChildrenByType(comp_node.INTEGER_
FEATURE);
    traverseTree(comp_node.getChildren(),
                comp_node.INTEGER_FEATURE,
                intFeatList);
    iter = intFeatList.iterator();

    //add the integer features to the UI
    while(iter.hasNext())
    {
        intFeat = (IntegerFeature)iter.next();
        String name = intFeat.getName();
        int val = intFeat.getIntValue();
        uiList.add("Integer Feature: " + name + " - " + val);
    }

    //get all the text features // line 167
    textFeatList = comp_node.getChildrenByType(comp_node.TEXT_FEATURE);
    traverseTree(comp_node.getChildren(),
                comp_node.TEXT_FEATURE,
                textFeatList);
    iter = textFeatList.iterator();

    //add the text features to the UI
```

```

        while(iter.hasNext())
        {
            textFeat = (TextFeature)iter.next();
            String name = textFeat.getName();
            String str = textFeat.getTextValue();
            uiList.add("Text Feature: " + name + " - " + str);
        }

        f.setSize(200,200);
        f.show();
    }
    catch(Exception e){e.printStackTrace();}

    return null;
}

/**
 * This function is used by generateOutput() to run through the config tree
and
 * pull out all of the items of a specified type. It is not part of the
FunctionalCompanion API,
 * but was written for this specific companion.
 * @param children this is a list of all the children of the current node
 * @param type this is the type we are currently searching for
 * @param resultList all items of the specified type which are found are
added to this list
 */
private void traverseTree(com.sun.java.util.collections.List children,
                        int type,
                        com.sun.java.util.collections.List resultList)
{
    if(!children.isEmpty())
    {
        Iterator iter = children.iterator();

        while(iter.hasNext())
        {
            RuntimeNode rtn = (RuntimeNode)iter.next();
            resultList.addAll(rtn.getChildrenByType(type));
            traverseTree(rtn.getChildren(),
                        type,
                        resultList);
        }
    }
}

```

```
}
```

Notes on the example

Line 2

```
import com.sun.java.util.collections.List; // line 2
```

If you are using JDK 1.1.x, import `com.sun.java.util.collections.List`, which is provided in `cz3rdpty.jar` (see [Section 1.3.2, "Installation Requirements for Java Functional Companions"](#) on page 1-7). If you are using JDK 1.2, then import `java.util.List`.

Line 35

```
super.initialize(comp_node, name, description, id); // line 35
```

In the `initialize()` method, call `super.initialize()`. This passes some of the necessary variables to the superclass so that its methods will work.

Lines 61-68

```
//get the necessary components from the configuration // line 61
```

This block illustrates how to get the logical state of an Option (with `getState()`), test the logical state (with the expression `== IState.UTRUE`), and set the value of a Feature (with `setIntValue()`).

Lines 102-109

```
//check to see if the value in the config is not at least the min value // line 102
```

This block produces the Configuration Status message shown in [Figure 1-3, "Testing Functional Companions in the Oracle SellingPoint application."](#)

4.3 Thin-Client generateOutput() Functional Companion

This Functional Companion uses the "thin-client" version of `generateOutput()` (see [Section 2.9.4](#) on page 2-27). When you invoke the Functional Companion from a web browser, it produces an HTML representation of the runtime Model tree, beginning at the node to which the companion is attached.

In order to use this type of Functional Companion, you must use Oracle Configurator in a web deployment. See the *Oracle Configurator Custom Web Deployment Guide* for details not covered in this document. Here is a summary of the tasks:

- Compile the Java source code into a class file.
- In Configurator Developer, define a Functional Companion rule with these options:

Option	Choice
Type	Output
Base Component	the Component to which you want to attach the Functional Companion
Implementation language	Java
Program String	the name of the class file

- In Configurator Developer's User Interface module, define a button for the Component that invokes the Functional Companion.
- With your internet server, create an OC servlet.
- Add the new class file for the Functional Companion to the CLASSPATH environment variable for the servlet.
- You can test the Functional Companion from Configurator Developer, by specifying the URL of the servlet (in Tools>Options>Test>Servlet URL) and clicking the Test button. This opens a web browser, passing it a URL that includes an XMLmsg parameter containing the necessary OC initialization message. This message contains database connection and login strings, and specifies the Model to display, by means of the `ui_def_id` parameter that identifies the User Interface definition you created in Configurator Developer.
- You can test the Functional Companion outside Configurator Developer, by creating an HTML test page that substitutes for your host application. (Examples are provided in the *Oracle Configurator Custom Web Deployment Guide*.) This page sends an OC initialization message that specifies database connection and login information, and the Model containing the Component. You can copy these parameters from the URL produced by the Test button in Configurator Developer. Test the Functional Companion by opening the HTML test page.

The example first calls the `response.setContentType()` method of the `HttpServletResponse` class, passing "text/html" as the output type. Then it calls `response.getWriter()` to get an output stream to which the Functional Companion can write HTML.

You can also write non-HTML output by setting a different content type (a MIME type) and writing appropriate data to the output stream.

Example 4-3 Thin-client Output Functional Companion

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.HttpServletResponse;
import com.sun.java.util.collections.Iterator;
import oracle.apps.cz.cio.FunctionalCompanion;
import oracle.apps.cz.cio.IRuntimeNode;

public class ShowStructure extends FunctionalCompanion {

    public void generateOutput(HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Runtime Model Structure</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>Runtime Model Structure</h3>");
        IRuntimeNode rootNode = getRuntimeNode();
        generateNode(out, rootNode, 0);
        out.println("</body>");
        out.println("</html>");
    }

    private static void generateNode(PrintWriter out, IRuntimeNode node, int
level) throws IOException {
        for (int i = 0; i < level; ++i) {
            out.print("--");
        }
        out.println(node.getName() + " <br> ");
        for (Iterator i = node.getChildren().iterator(); i.hasNext(); ) {
            IRuntimeNode childNode = (IRuntimeNode)i.next();
            generateNode(out, childNode, (level + 1));
        }
    }
}
```

}

Glossary of Terms

This glossary for Oracle Configurator is followed by a Glossary of Acronyms

Active Model

The part of Oracle Configurator runtime architecture that processes model structure and rules to create configurations. Interfaces dynamically with the end user Active UI and data.

Active User Interface

The part of Oracle Configurator runtime architecture that provides the graphical views necessary to create configurations interactively. Interfaces with the Active Model and data to give users access to customer requirements gathering, product selection, and customer-centric extensions.

Application Architecture

The software structure of an application at runtime. Architecture affects how an application is used, maintained, extended, and changed.

Architecture

The software structure of a system. Architecture affects how a system is used, maintained, extended, and changed. See also Application Architecture.

Beta

An external release, delivered as an installable application, and subject to system, validation, and acceptance testing. Specially selected and prepared end users may participate in beta testing.

Bill of Material

A list of component items associated with a parent item (assembly) and information about how each item relates to the parent item.

BOM

See Bill of Material.

BOM Item

The nodes imported into the Oracle Configurator Developer Model that correspond to an Oracle BOM.

BOM Model

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Model in an Oracle BOM.

BOM OptionClass

The imported Model node in the Oracle Configurator Developer that corresponds to Option Class in an Oracle BOM.

BOM StandardItem

The imported Model node in the Oracle Configurator Developer that corresponds to Standard Item in an Oracle BOM.

Boolean Expression

An element of a component in the Model that has two options: true or false.

Bug

See Defect.

Build

A specific instance of an application during its construction. A build must have an install early in the project so that application implementers can unit test their latest work in the context of the entire available application.

CIO

See Oracle Configuration Interface Object.

Client

A runtime program using a server to access functionality shared with other clients.

Comparison Rule

An Oracle Configurator Developer rule type to establish a relationship that determines the selection state of a logical item (option, boolean feature, or list-of-options feature) based on a comparison of two numeric values (numeric features, totals, resources, 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 to establish a relationship among features in the Model that specifies the allowable combinations of options. See also, Property-based Compatibility Rule.

Compatibility Table

A type of compatibility relationship where the allowable combination of options are explicitly enumerated.

Component

Represents a configurable element in a product. An element of the Model structure, typically containing features. May correspond to one screen of selections in an Oracle runtime configurator.

Component Set

An element of the Model that contains a number of components of the same type, where each component of the set is independently configured.

Configuration

A specific set of specifications for a product, resulting from selections made in an Oracle runtime configurator.

Configuration Model

The model structure and rules-based content of an Oracle runtime configurator. The configuration model is constructed and maintained using Oracle Configurator Developer, and is interpreted at runtime by the Active Model.

Configuration Rules

The Oracle Configurator Developer logic rules and numeric rules available for defining configurations.

Configurator

The part of an application that provides custom configuration capabilities.

Constraint Rule

An Oracle Configurator Developer rule type to create a logical relationship among features and options. See also Rules.

Contributes to

An Oracle Configurator Developer numeric rule type for accumulating a total value.

Consumes from

An Oracle Configurator Developer numeric rule type for specifying the quantity of a resource used.

CRM

Customer Relationship Management. The aspect of the enterprise that involves contact with customers, from lead generation to support services.

Customer

The person or persons for whom products are configured by end users of the Oracle Configurator or other ERP and CRM applications.

Customer-centric Extensions

See Customer-centric Views.

Customer-centric Views

Optional extensions to core functionality that supplement product selection with rules for pre-selection, validation, and intelligent views. View capabilities include generative geometry, drawings, sketches and schematics, charts, performance analyses, and ROI calculations.

Customer Requirements

The needs of the customer that serve as the basis for determining the configuration of products, systems, and/or services. Also called Needs Assessment.

Data Import

Populating the Oracle Configurator schema with enterprise data from ERP or legacy systems via import tables.

Data Integration Object

Data Integration Object. A server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the Oracle Configurator schema.

Data Maintenance Environment

The environment in which the Oracle runtime configurator data is maintained.

Data Replication

The activity of downloading and uploading configuration, quote, and order data between the Oracle Configurator schema on the enterprise server and Oracle Configurator Mobile Database on end-user mobile laptop PCs. See also Data Synchronization.

Datasource

A programmatic reference to a database. Referred to by a datasource name, or DSN.

Data Synchronization

A process for matching the data in the Oracle Configurator schema and the data available to client processes such as the Oracle SellingPoint application. See also Data Replication.

Default

The automatic selection of an option based on the pre-selection rules or the selection of another option.

Defaults

An Oracle Configurator Developer logic rule to determine the logic state of features or options in a default relation to other features and options. For instance, if you set A to True by selecting it, B becomes true (selected) if it is available (not false) and can be set to True without contradicting a non-default rule or a previous default setting for B.

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".

Defect Tracking

A system of identifying defects for managing additional tests, testing, and approval for release to users.

Deliverable

A work product that is specified for review and delivery.

Demonstration

A presentation of the tested application, showing a particular usage scenario.

Design Chart

An Oracle Configurator Developer rule type for defining advanced Explicit Compatibilities interactively in a chart view.

Design Review

A technical review that focuses on application or system design.

Developer

The tool (Oracle Configurator Developer) used to create configuration models. The person who uses Oracle Configurator Developer to create a configurator. See also Implementer

DIO

See Data Integration Object.

End User

The ultimate user of the Oracle runtime 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 web or kiosk.

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.

Excludes

An Oracle Configurator Developer rule type for determining the logic state of features or options in an excluding relation to other features and options. For instance, if you set A to True, B becomes false, since it is not allowed when A is true. If you set A to False, there is no effect on B, meaning it could be true, false, or unknown.

Extended Functionality

A release after delivery of core functionality that extends that core functionality with customer-centric views, more complex proposal generation, discounting, quoting, and expanded integration with ERP, CRM, and third-party software.

Feature

An element of the Model structure. A configurable parameter of a component. Features can either have a value (numeric or boolean) or enumerated options.

Functional Companion

An object associated with a component that supplies methods that can be used to initialize, validate and generate customer-centric views and outputs for the configuration.

Functional Specification

Document describing the functionality of the application based on user requirements.

Incremental Construction

The process of organizing the construction of the application into builds, where each build is designed to meet a specified portion of the overall requirements and is unit tested.

Implementation

The stage in a project between defining the problem by selecting a configuration technology vendor, such as Oracle, and deploying the completed sales configuration application. The Implementation stage includes gathering requirements, defining test cases, designing the application, constructing and testing the application, and delivering it to users.

Implementer

The person who uses Oracle Configurator Developer to build the model structure, rules, and UI customizations that make up an Oracle runtime configurator.

Implies

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 instance, if you set A to True by selecting it, B becomes true, since selecting A implies that B is also selected. If you set A to False by deselecting it, there is no effect on B, meaning it could be true false or unknown based on other relations B participates in. And if you set B to True by selecting it, there is no effect on A, meaning it could be true false or unknown based on other relations A participates in. But if you set B to False by deselecting it, the relation of A implies B is preserved only by having A be false (deselected) as well.

Import Tables

Tables mirroring the Oracle Configurator schema Item Master structure, but without integrity constraints. Import Tables allow batch population of the Oracle Configurator schema Item Master. Import Tables are used in conjunction with extractions from Oracle Applications or legacy data to create, update, or delete records in the Oracle Configurator schema Item Master.

Install

A program that sets up the local machine and installs the application for testing and use.

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.

Intelligent Views

Configuration output, such as reports, graphs, schematics, and diagrams, that help to illustrate the value proposition of what is being sold.

Item Master

A table in the Oracle Configurator schema containing data used to structure the product. Data in the item master is either entered manually or imported from Oracle Applications or legacy data.

Item Type

A table in the Oracle Configurator schema containing data used to classify the product data in the item master table.

Log File

A file containing errors, warnings and other information output by the running application.

Logic Rules

Logic rules directly or indirectly set the logical state (true, false, or unknown) of features and options in the Model.

There are four (4) primary logic rules: Implies, Requires, Excludes, and Negates. Each of these rules takes a list of features or options as operands. See also Logic, Implies, Requires, Excludes, and Negates.

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 bug fixes, procedure changes, infrastructure adjustments, data replication schedules, etc.

Maintenance Guide

A guide for maintaining a specific application or system. The maintenance guide covers all aspects of maintenance described in the generic Maintenance Plan.

Maintenance Plan

A document that outlines what is required for successful maintenance, and who is responsible for all the actions and deliverables of carrying out maintenance on a system.

MDUI

See Model-driven UI.

Mobile Database

See Oracle Configurator Mobile Database.

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. May consist of BOM Items.

Model-driven UI

The graphical views of the model structure and rules generated by the Active UI to present end users with interactive product selection based on configuration models.

Model Structure

Hierarchical, “tree” view of data in terms of product elements (Models, Products Components, Features, Options, BOM Models, BOM OptionClasses, BOM StandardItems, Resources, and Totals). May include reusable components.

MRP

Manufacturing Resource Planning. A software system and process for monitoring and maintaining the customer's manufacturing systems.

Negates

An 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 instance, if you set one item in the relationship to True, the other item must be false. And if you set one item to False, the other item must be true.

Node

The place in a Model occupied by a component, feature, option or variable, BOM Model, BOM OptionClass, or BOM StandardItem.

Numeric Rules

Rules that are used to set the global parameters specified in product structuring. See also, Contributes to and Consumes from.

OC

See Oracle Configurator.

Opportunity

The workspace in the Oracle SellingPoint application and Oracle Sales Online in which products, systems, and/or services are configured, quotes and proposals are generated, and orders are submitted.

Option

An element of the Model. A choice for the value of an enumerated feature.

A logical selection made by the end user when configuring a component.

Oracle Configurator

The product family consisting of development tools and runtime applications such as Oracle Configurator schema, Oracle Configurator Developer, Oracle Configurator window, and Oracle SellingPoint application. Also the Oracle runtime configurator variously packaged for use in networked, mobile, or web deployments.

Oracle Configurator Architecture

The application runtime architecture consists of the Active User Interface, the Active Model, and the Oracle Configurator schema or Oracle Configurator Mobile Database. The application development architecture consists of Oracle Configurator Developer and the Oracle Configurator schema, with test instances of an Oracle runtime configurator.

Oracle Configurator Developer

The suite of tools in the Oracle Configurator product family for constructing and maintaining configurators.

Oracle Configuration Interface Object (CIO)

A server in the runtime application that creates and manages the interface between the client (usually a user interface like the Active User Interface) and the underlying representation of model structure and rules in the Active Model.

CIO protocols support creating and navigating the Model, querying and modifying selection states, and saving and restoring configurations.

Oracle Configurator Mobile Database

The runtime version of the standard Oracle Configurator schema that manages data for the configuration model in a mobile deployment. The runtime schema includes customer, product, and pricing data as well as data created during operation of an Oracle Configurator.

Oracle Configurator Schema

The implementation version of the standard Oracle runtime 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.

Oracle SellingPoint Application

The test application generated by Oracle Configurator Developer. Also a full configuration environment with opportunity management, quotes, and proposals for networked or mobile deployments.

Output

The output generated by a configurator, such as quotes, proposals, bills of material (BOM), and customer-centric views.

PDM

Product Data Management. A software system that manages the version control of product data.

Populator

An entity in the Oracle Configurator Developer that defines how to create a Model from information in the item master.

Pre-selection

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.

Principal Design Consultant

Member of the project team responsible for architecting the design of the application.

Product

Whatever is subjected to configuration and is the output of the application.

The root element of the Model.

Product Family

A collection of products or product lines, which are organized as a group by a provider or manufacturer.

Project

The workspace in Oracle Configurator Developer in which configurators are constructed

Project Manager

A member of the project team who is responsible for directing the project during implementation.

Project Plan

A document that outlines the logistics of successfully implementing the project, including the schedule.

Property

A named value associated with an object in the Model or the item master. A set of properties may be associated with an item type.

Property-based Compatibility Rule

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, to prove feasibility or examine other implementation issues.

Reference

The use of a reusable component within the Model. Not implemented in Release 11i or before.

Regression Test

An automated test that ensures the newest build still meets previously tested requirements and functionality.

Requires

An Oracle Configurator Developer logic rule type that determines the logic state of features or options in a requirement relation to other features and options. For instance, if you set one item in the relationship to True, the other item is required to be true as well. And if you set one item to False, the other item must be false as well.

Resource

Staff or materials available or needed within an enterprise.

A variable in the Model used to maintain the balance of features not consuming more of a specific resource than has been provided by other features.

Reusable Component

A component that is referenced from multiple locations in the Model. Not implemented in Release 11*i* or before.

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 rules. 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 are used to provide pre-selection and validation capability in an application.

See also Logic Rules and Numeric Rules.

Runtime

The environment and context in which applications are run or used, rather than developed.

Sales Configuration

A part of the sales process to which configuration technology has been applied in order to increase sales effectiveness and decrease order errors. Commonly identifies needs assessment and product configuration.

Server

Centrally located software processes or hardware, shared by clients.

Solution

The deployed system as a response to a problem or problems.

System

The hardware and software components and infrastructure integrated to satisfy functional and performance requirements.

Test Case

A description of inputs, execution instructions, and expected results, which are created for the purpose of determining whether a specific software feature works correctly or a specific requirement has been met.

Total

A variable in the Model used to accumulate a numeric total, such as total price or total weight.

Undetermined

The logic state that is neither true nor false, but unknown at the time a logic rule is executed. This logic state is also referred to as Available, especially when considered from the point of view of the Oracle runtime configurator end user.

Unit Test

Execution of individual routines and modules by the application implementer or by an independent test consultant for the purposes of finding defects.

Update

Moving a production configurator to a new version of configuration model.

Upgrade

Moving the configurator to a new release of Oracle Configurator.

User

The person using the Oracle Configurator Developer tools and methods to build an Oracle runtime configurator. See also end user.

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.

User Requirements

A description of what the Oracle Configurator or Oracle SellingPoint application is expected to do from the end user's perspective.

User's Guide

Documentation on using the application or configurator to solve the intended problem.

Validation

Tests that ensure that the configured components will meet specific performance or acceptance criteria.

A type of functional companion that is implemented to ensure that the configured components will meet specific performance or acceptance criteria.

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.

Glossary of Acronyms

API

Application Programming Interface

ATP

Available to Promise

BOM

Bill of Material

CIO

Configuration Interface Object

CM

Configuration Management

COM

Component Object Model

CRM

Customer Relationship Management

DBMS

Database Management System

DCOM

Distributed Component Object Modeling

DHTML

Dynamic Hypertext Markup Language

DIO

Data Integration Object

DLL

Dynamically Linked Library

DXF

Drawing Exchange Format (AutoCAD drawings)

ECO

Engineering Change Order

ERM

Enterprise Relationship Management

ERP

Enterprise Resource Planning

ESD

Electronic Software Distribution

ESP

External Service Provider

ESS

Enterprise Selling System

HT

High Tech

HTML

Hypertext Markup Language

IP

Industrial Products

IS

Information Services

ISS

Interactive Selling System

ISV

Independent Software Vendor

LAN

Local Area Network

MAPI

Messaging Application Programming Interface

MC/S

Mobile Client/Server System

MDUI

Model-Driven User Interface

MES

Marketing Encyclopedia System (Catalog)

MIS

Management Information Systems

MRP

Manufacturing Resource Planning

MS

Microsoft

OC

Oracle Configurator

OCX

Object Control File, OLE custom controls

ODBC

Open Database Connectivity

OLE

Object linking and embedding

OMS

Opportunity Management System

OOD

Object-Oriented Design

ORB

Object Request Broker

PDM

Product Data Management

PIA

Project Impact Assessment

POS

Point of Sale

QA

Quality Assurance

RAD

Rapid Application Development

RDBMS

Relational Database Management System

RFQ

Request for Quote

ROI

Return on Investment

SAS

Sales Analysis System

SCM

Supply Chain Management

SCS

Sales Configuration System

SE

Sales Engineer

SFA

Sales Force Automation

SI

System Integrator

SOT

Strategic Options Theory

SQA

Software Quality Assurance

SQL

Structured Query Language

TERM

Technology-Enabled Relationship Management

TES

Technology-Enabled Selling

UI

User Interface

VAR

Value-Added Reseller

VB

Microsoft Visual Basic

WAN

Wide Area Network

WIP

Work In Progress

CIO Package and Related Classes

Package Summary

Packages

[Package oracle.apps.cz.cio](#) Provides classes used to create, save and restore configurations.

[Package oracle.apps.cz.common](#)

[Package oracle.apps.cz.utilities](#)

Package oracle.apps.cio

Description

Provides classes used to create, save and restore configurations. The top-level entry point to this package is the configuration integration object, which provides methods for creating, saving, restoring and deleting configurations. The configuration can be manipulated by calling methods on the configuration object and its tree of runtime objects.

Class Summary

Interfaces

IAtp	Implemented by objects that can have ATP calculated.
IBomItem	Implemented by all selectable BOM items.
ICompSetEventListener	Implemented by objects that want to find out about added components.
IConfigEventListener	Implemented by objects that want to find out about added components.
ICount	Implemented by objects that have an associated integer count.
IDecimal	Implemented by objects that can both get and set a decimal value.
IDecimalMinMax	Implemented by objects that have a decimal minimum and maximum value.
IFunctionalCompanion	Implemented by functional companion objects attached to components in order to provide programatic functionality to a configuration model.
IInteger	Implemented by objects that have an integer value.
IIntegerMinMax	Implemented by objects that have an integer minimum and maximum.
IOption	Implemented by objects that act as options.

Class Summary

IOptionFeature	Implemented by objects that contain selectable options.
IPrice	Implemented by objects that can be priced.
IReadOnlyDecimal	Implemented by objects that have a decimal value.
IRuntimeNode	Implemented by all objects in the runtime configuration tree.
IState	Implemented by objects that have logic state.
IText	Implemented by objects that have a textual value.
Classes	
AutoFunctionalCompanion	Base object on which user Functional Companions can be based.
BomModel	Represents configurable BOM Models.
BomNode	Represents configurable BOM option classes.
BomOptionClass	Represents configurable BOM option classes.
BomStdItem	Represents configurable BOM models.
BooleanFeature	Represents a feature with a boolean value.
CIO	Implements a configuration integration object that can be used to create, save, restore and delete configurations.
CompanionNode	Provides methods for extracting values from a model based on property annotations.
CompanionRoot	Provides functional companion implementors with property-based feature extraction.
CompanionValidationFailure	Failure produced by a functional companion object.
Component	Implements the runtime configuration behavior of products and components.
ComponentNode	Represents a set of configurable components.
ComponentSet	Represents a set of configurable components.
ConfigTransaction	Represents a configuration transaction.
Configuration	The top-level entry point to a configuration.
CountFeature	Represents a countable integer feature.
DecimalFeature	Represents a feature with a decimal value.
DecimalNode	An abstract class implementing behavior common to objects with a decimal value.

Class Summary

Factory	Provides a class factory for the CIO to clients that do not support arguments to constructors.
FunctionalCompanion	Base object on which user functional companions can be based.
FunctionalCompanionException	This exception is used to indicate that an error occurred somewhere inside the functional companion.
IntegerFeature	Represents a feature with an integer value.
IntegerNode	Represents a feature with an integer value.
Option	Represents an option of an option feature.
OptionFeature	Represents a feature with selectable options.
OptionFeatureNode	An abstract class implementing behavior common to all features with options.
OptionNode	An abstract class implementing behavior common to all option-like objects.
Property	Represents name/value properties associated with runtime nodes.
ReadOnlyDecimalNode	An abstract class implementing behavior common to objects with a decimal value.
Reason	This class wraps the information returned by a contradiction in order to include information about internal error messages.
Resource	Represents a consumable resource.
RestoreValidationFailure	Failure produced when restoring a configuration over a changed model.
RuntimeNode	Abstract class implementing common behavior across all runtime nodes.
StateCountNode	Abstract class implementing common behavior for nodes with a logic state and count.
StateNode	Abstract class implementing common behavior across nodes with logic state.
StatusInfo	Contains information about a status change for a particular runtime node.
TextFeature	Represents a feature that has a textual value.
TextNode	Represents a feature that has a textual value.
Total	Represents a total that has a decimal numeric value.
ValidationFailure	Implements behavior common to all validation failures.
Exceptions	
AtpUnavailableException	Signals that the CIO ATP calculation functionality is not available.

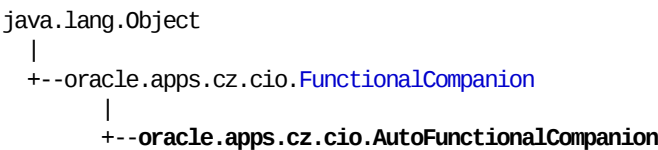
Class Summary

BomExplosionException	Exception which is thrown when a client tries to create a configuration directly from an Apps bill of material and there is a problem with the explosion of the bill.
FuncCompCreationException	Signalled if a functional companion cannot be created.
FuncCompMessageException	FuncCompMessageException is designed to be thrown from a functional companion's autoConfigure() method code when the author of the companion wants to display a BENIGN message in a dialog box to the end user.
IncompatibleInputException	Signalled if a particular input is of different type than the node it is trying to restore over.
LogicalException	Signalled if a logical failure occurs.
LogicalOverridableException	Signalled if a logical contradiction occurs that can be overridden.
LogicalRuntimeException	Signalled if a fatal logic exception occurred.
MissingFileException	Signalled if a particular logic file is missing.
MissingLogicException	Signalled if a particular logic record is missing.
NoAtpCalculatedException	Exception which is thrown when an ATP method is called on an item for which ATP is not calculated.
NoConfigHeaderException	Signalled if the configuration hasn't been saved yet.
NonPricedNodeException	Exception which is thrown when a pricing method is called on an item which should not be priced.
NoSuchChildException	Signalled if a requested child does not exist.
NotOneProductException	Exception which is thrown when a client tries to create a configuration by specifying the name of the project and the project contains more than one or no products.
NotOneProjectException	Exception which is thrown when a client tries to create a configuration by specifying the name of the project and the project name identifies more than one or no projects.
PricingUnavailableException	Signals that the CIO pricing functionality is not available.
SelectionNotMutexedException	Signalled when an mutexed selection operation is performed on an option feature that does not support mutexed selection.
TransactionException	Signalled if a particular logic file is missing.

oracle.apps.cz.cio AutoFunctionalCompanion

Syntax

public class AutoFunctionalCompanion extends [FunctionalCompanion](#)



All Implemented Interfaces:

[IFunctionalCompanion](#)

Description

Base object on which user Functional Companions can be based.

Provides the ability to modify ("side-effect") a configuration Model when a configuration instance is initialized or saved. (This functionality is not allowed by [FunctionalCompanion](#) and its interface class [IFunctionalCompanion](#).) To use this functionality, you extend your own Functional Companion class from [AutoFunctionalCompanion](#).

Member Summary

Constructors

[AutoFunctionalCompanion\(\)](#)

Methods

onNew()	Does nothing.
onRestore()	Does nothing.
onSave()	Does nothing.

Constructors

AutoFunctionalCompanion()

```
public AutoFunctionalCompanion()
```

Methods

onNew()

```
public void onNew()
```

Does nothing. Called right after the activation of a newly-created configuration.

onRestore()

```
public void onRestore()
```

Does nothing. Called after restoring a configuration.

onSave()

```
public void onSave()
```

Does nothing. Called before saving a configuration.

oracle.apps.cz.cio AtpUnavailableException

Syntax

```
public class AtpUnavailableException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.AtpUnavailableException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Signals that the CIO ATP calculation functionality is not available.

Member Summary

Constructors

[AtpUnavailableException\(String\)](#)

[AtpUnavailableException\(String,
Object, Log\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

[fillInStackTrace](#), [getLocalizedMessage](#), [getMessage](#), [printStackTrace](#), [printStackTrace](#), [printStackTrace](#), [toString](#)

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Constructors

AtpUnavailableException(String)

```
public AtpUnavailableException(java.lang.String reason)
```

AtpUnavailableException(String, Object, Log)

```
public AtpUnavailableException(java.lang.String reason, java.lang.Object  
source, oracle.apps.fnd.common.Log log)
```

oracle.apps.cz.cio BomExplosionException

Syntax

```
public class BomExplosionException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.BomExplosionException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Exception which is thrown when a client tries to create a configuration directly from an Apps bill of material and there is a problem with the explosion of the bill.

Member Summary

Methods

[getExplosionDate\(\)](#)

[getInventoryItemId\(\)](#)

[getOrganizationId\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

[fillInStackTrace](#), [getLocalizedMessage](#), [getMessage](#), [printStackTrace](#), [printStackTrace](#), [printStackTrace](#), [toString](#)

Methods inherited from class java.lang.Object

Inherited Member Summary

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods

getExplosionDate()

```
public java.util.Date getExplosionDate()
```

getInventoryItemId()

```
public int getInventoryItemId()
```

getOrganizationId()

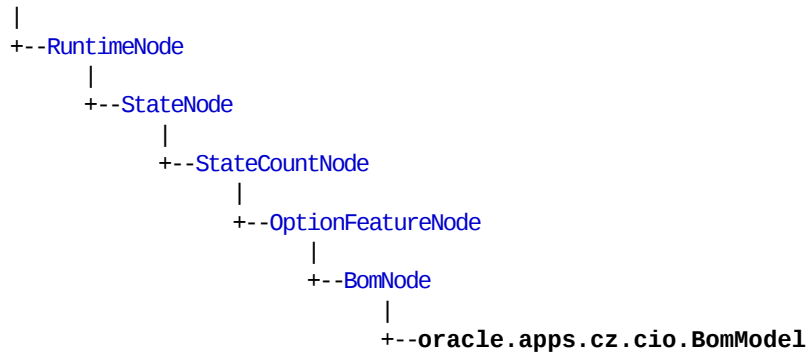
```
public int getOrganizationId()
```

oracle.apps.cz.cio BomModel

Syntax

public class BomModel extends [BomNode](#)

java.lang.Object



All Implemented Interfaces:

[IAtp](#), [IBomItem](#), [ICount](#), [IOption](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents configurable BOM Models.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [BomNode](#)

[addAtpNotification\(String\)](#), [areOptionsCounted\(\)](#), [calculateAtpDate\(\)](#), [clearAtpDate\(\)](#), [clearAtpNotifications\(\)](#), [deselect\(\)](#), [getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getComponentCode\(\)](#), [getDefaultQuantity\(\)](#), [getDiscountedPrice\(\)](#), [getInventoryItemId\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getMaxQuantity\(\)](#), [getMaxSelected\(\)](#), [getMinQuantity\(\)](#), [getMinSelected\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [getState\(\)](#), [getUomCode\(\)](#), [hasDefaultQuantity\(\)](#), [hasMaxQuantity\(\)](#), [hasMaxSelected\(\)](#), [hasMinQuantity\(\)](#), [hasMinSelected\(\)](#), [isOptionMutexed\(\)](#), [isRequired\(\)](#), [isSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(\)](#), [select\(IOption\)](#), [setAtpDate\(Date\)](#), [setState\(int\)](#)

Methods inherited from class [OptionFeatureNode](#)

[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getExtendedPrice\(\)](#), [getPricingNotifications\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildById\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseId\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeId\(\)](#), [getSelectionLineId\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineId\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals\(\)](#), [getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(\)](#), [wait\(\)](#)

Methods inherited from interface [IBomItem](#)

[getComponentCode\(\)](#), [getInventoryItemId\(\)](#), [getMaxQuantity\(\)](#), [getMinQuantity\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [hasMaxQuantity\(\)](#), [hasMinQuantity\(\)](#), [isRequired\(\)](#)

Methods inherited from interface [IOptionFeature](#)

[areOptionsCounted\(\)](#), [deselect\(IOption\)](#), [getMaxSelected\(\)](#), [getMinSelected\(\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [hasMaxSelected\(\)](#), [hasMinSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from interface [IState](#)

Inherited Member Summary

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IOption](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IAtp](#)

[getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getDatabaseID\(\)](#), [getItemKey\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

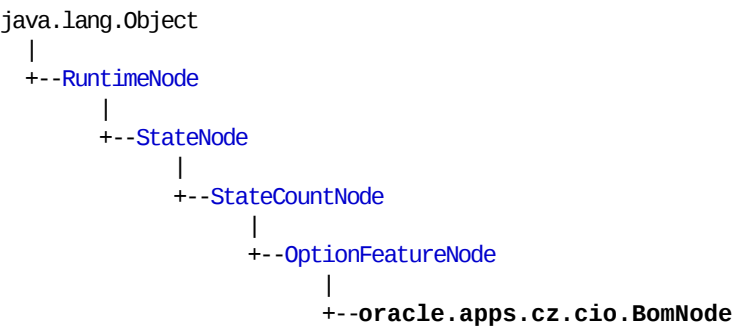
[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio

BomNode

Syntax

public abstract class BomNode extends [OptionFeatureNode](#) implements [IBomItem](#)



Direct Known Subclasses:

[BomModel](#), [BomOptionClass](#), [BomStdItem](#)

All Implemented Interfaces:

[IAtp](#), [IBomItem](#), [ICount](#), [IOption](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents configurable BOM option classes.

Member Summary

Methods

- [addAtpNotification\(String\)](#)
- [areOptionsCounted\(\)](#)
- [calculateAtpDate\(\)](#)
- [clearAtpDate\(\)](#)
- [clearAtpNotifications\(\)](#)

Member Summary

deselect()
getAtpDate()
getAtpNotifications()
getComponentCode()
getDefaultQuantity()
getDiscountedPrice()
getInventoryItemId()
getItemKey()
getListPrice()
getMaxQuantity()
getMaxSelected()
getMinQuantity()
getMinSelected()
getOrganizationId()
getPrimaryUomCode()
getState()
getUomCode()
hasDefaultQuantity()
hasMaxQuantity()
hasMaxSelected()
hasMinQuantity()
hasMinSelected()
isOptionMutexed()
isRequired()
isSelected()
isSelectionMutexed()

Member Summary

[select\(\)](#)
[select\(IOption\)](#)
[setAtpDate\(Date\)](#)
[setState\(int\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)
[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [LTRUE](#)

Fields inherited from interface [IRuntimeNode](#)
[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [OptionFeatureNode](#)
[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#)

Methods inherited from class [StateCountNode](#)
[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getExtendedPrice\(\)](#), [getPricingNotifications\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)
[isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)
[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)
[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IOptionFeature](#)
[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#)

Inherited Member Summary

Methods inherited from interface [IState](#)

[isUnknown\(\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getExtendedPrice\(\)](#), [getPricingNotifications\(\)](#)

Methods inherited from interface [IAtp](#)

[getDatabaseID\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

addAtpNotification(String)

```
public void addAtpNotification(java.lang.String message)
```

areOptionsCounted()

```
public boolean areOptionsCounted()
```

Specified By:

[areOptionsCounted\(\)](#) in interface [IOptionFeature](#)

Specified By:

[areOptionsCounted\(\)](#) in interface [IOptionFeature](#)

calculateAtpDate()

```
public void calculateAtpDate()
```

clearAtpDate()

```
public void clearAtpDate()
```

clearAtpNotifications()

```
public void clearAtpNotifications()
```

deselect()

```
public void deselect()
```

Specified By:

[deselect\(\)](#) in interface [IOption](#)

getAtpDate()

```
public java.util.Date getAtpDate()
```

Specified By:

[getAtpDate\(\)](#) in interface [IAtp](#)

getAtpNotifications()

```
public java.lang.String getAtpNotifications()
```

Specified By:

[getAtpNotifications\(\)](#) in interface [IAtp](#)

getComponentCode()

```
public java.lang.String getComponentCode()
```

Specified By:

[getComponentCode\(\)](#) in interface [IBomItem](#)

getDefaultQuantity()

```
public int getDefaultQuantity()
```

getDiscountedPrice()

```
public double getDiscountedPrice()
```

Specified By:

[getDiscountedPrice\(\)](#) in interface [IPrice](#)

Specified By:

[getDiscountedPrice\(\)](#) in interface [IPrice](#)

Overrides:

[getDiscountedPrice\(\)](#) in class [StateCountNode](#)

getInventoryItemId()

public int getInventoryItemId()

Specified By:

[getInventoryItemId\(\)](#) in interface [IBomItem](#)

getItemKey()

public java.lang.String getItemKey()

Specified By:

[getItemKey\(\)](#) in interface [IPrice](#)

Specified By:

[getItemKey\(\)](#) in interface [IAtp](#)

Specified By:

[getItemKey\(\)](#) in interface [IPrice](#)

Overrides:

[getItemKey\(\)](#) in class [StateCountNode](#)

getListPrice()

public double getListPrice()

Specified By:

[getListPrice\(\)](#) in interface [IPrice](#)

Specified By:

[getListPrice\(\)](#) in interface [IPrice](#)

Overrides:

[getListPrice\(\)](#) in class [StateCountNode](#)

getMaxQuantity()

```
public int getMaxQuantity()
```

Specified By:

[getMaxQuantity\(\)](#) in interface [IBomItem](#)

getMaxSelected()

```
public int getMaxSelected()
```

Specified By:

[getMaxSelected\(\)](#) in interface [IOptionFeature](#)

Specified By:

[getMaxSelected\(\)](#) in interface [IOptionFeature](#)

getMinQuantity()

```
public int getMinQuantity()
```

Specified By:

[getMinQuantity\(\)](#) in interface [IBomItem](#)

getMinSelected()

```
public int getMinSelected()
```

Specified By:

[getMinSelected\(\)](#) in interface [IOptionFeature](#)

Specified By:

[getMinSelected\(\)](#) in interface [IOptionFeature](#)

getOrganizationId()

```
public int getOrganizationId()
```

Specified By:

[getOrganizationId\(\)](#) in interface [IBomItem](#)

getPrimaryUomCode()

```
public java.lang.String getPrimaryUomCode()
```

Specified By:

[getPrimaryUomCode\(\)](#) in interface [IBomItem](#)

getState()

```
public int getState()
```

Specified By:

[getState\(\)](#) in interface [IState](#)

Specified By:

[getState\(\)](#) in interface [IState](#)

Specified By:

[getState\(\)](#) in interface [IState](#)

Specified By:

[getState\(\)](#) in interface [IState](#)

Overrides:

[getState\(\)](#) in class [StateNode](#)

getUomCode()

```
public java.lang.String getUomCode()
```

Specified By:

[getUomCode\(\)](#) in interface [IPrice](#)

Specified By:

[getUomCode\(\)](#) in interface [IAtp](#)

Specified By:

[getUomCode\(\)](#) in interface [IPrice](#)

Overrides:

[getUomCode\(\)](#) in class [StateCountNode](#)

hasDefaultQuantity()

public boolean hasDefaultQuantity()

hasMaxQuantity()

public boolean hasMaxQuantity()

Specified By:

[hasMaxQuantity\(\)](#) in interface [IBomItem](#)

hasMaxSelected()

public boolean hasMaxSelected()

Specified By:

[hasMaxSelected\(\)](#) in interface [IOptionFeature](#)

Specified By:

[hasMaxSelected\(\)](#) in interface [IOptionFeature](#)

hasMinQuantity()

public boolean hasMinQuantity()

Specified By:

[hasMinQuantity\(\)](#) in interface [IBomItem](#)

hasMinSelected()

public boolean hasMinSelected()

Specified By:

[hasMinSelected\(\)](#) in interface [IOptionFeature](#)

Specified By:

[hasMinSelected\(\)](#) in interface [IOptionFeature](#)

isOptionMutexed()

public boolean isOptionMutexed()

Specified By:

[isOptionMutexed\(\)](#) in interface [IOption](#)

isRequired()

public boolean isRequired()

Specified By:

[isRequired\(\)](#) in interface [IBomItem](#)

isSelected()

public boolean isSelected()

Specified By:

[isSelected\(\)](#) in interface [IOption](#)

isSelectionMutexed()

public boolean isSelectionMutexed()

Specified By:

[isSelectionMutexed\(\)](#) in interface [IOptionFeature](#)

Specified By:

[isSelectionMutexed\(\)](#) in interface [IOptionFeature](#)

Overrides:

[isSelectionMutexed\(\)](#) in class [OptionFeatureNode](#)

select()

```
public void select()
```

Specified By:

[select\(\)](#) in interface [IOption](#)

select(IOption)

```
public void select(IOption option)
```

Specified By:

[select\(IOption\)](#) in interface [IOptionFeature](#)

Specified By:

[select\(IOption\)](#) in interface [IOptionFeature](#)

Overrides:

[select\(IOption\)](#) in class [OptionFeatureNode](#)

setAtpDate(Date)

```
public void setAtpDate(java.util.Date atpDate)
```

setState(int)

```
public void setState(int newState)
```

Specified By:

[setState\(int\)](#) in interface [IState](#)

Specified By:

[setState\(int\)](#) in interface [IState](#)

Specified By:

[setState\(int\)](#) in interface [IState](#)

Specified By:

[setState\(int\)](#) in interface [IState](#)

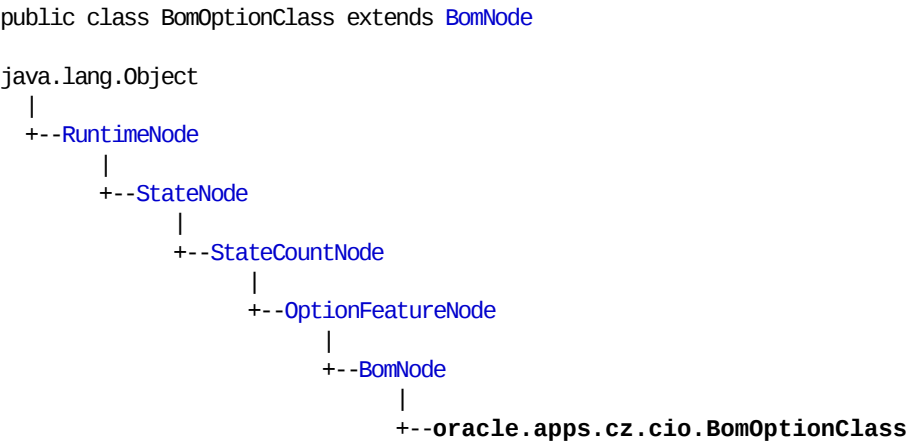
Overrides:

[setState\(int\)](#) in class [StateNode](#)

oracle.apps.cz.cio

BomOptionClass

Syntax



All Implemented Interfaces:

[IAtp](#), [IBomItem](#), [ICount](#), [IOption](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents configurable BOM option classes.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [BomNode](#)

[addAtpNotification\(String\)](#), [areOptionsCounted\(\)](#), [calculateAtpDate\(\)](#), [clearAtpDate\(\)](#), [clearAtpNotifications\(\)](#), [deselect\(\)](#), [getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getComponentCode\(\)](#), [getDefaultQuantity\(\)](#), [getDiscountedPrice\(\)](#), [getInventoryItemId\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getMaxQuantity\(\)](#), [getMaxSelected\(\)](#), [getMinQuantity\(\)](#), [getMinSelected\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [getState\(\)](#), [getUomCode\(\)](#), [hasDefaultQuantity\(\)](#), [hasMaxQuantity\(\)](#), [hasMaxSelected\(\)](#), [hasMinQuantity\(\)](#), [hasMinSelected\(\)](#), [isOptionMutexed\(\)](#), [isRequired\(\)](#), [isSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(\)](#), [select\(IOption\)](#), [setAtpDate\(Date\)](#), [setState\(int\)](#)

Methods inherited from class [OptionFeatureNode](#)

[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getExtendedPrice\(\)](#), [getPricingNotifications\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildById\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseId\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeId\(\)](#), [getSelectionLineId\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineId\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeId\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals\(\)](#), [getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(\)](#), [wait\(\)](#)

Methods inherited from interface [IBomItem](#)

[getComponentCode\(\)](#), [getInventoryItemId\(\)](#), [getMaxQuantity\(\)](#), [getMinQuantity\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [hasMaxQuantity\(\)](#), [hasMinQuantity\(\)](#), [isRequired\(\)](#)

Methods inherited from interface [IOptionFeature](#)

[areOptionsCounted\(\)](#), [deselect\(IOption\)](#), [getMaxSelected\(\)](#), [getMinSelected\(\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [hasMaxSelected\(\)](#), [hasMinSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from interface [IState](#)

Inherited Member Summary

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IOption](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IAtp](#)

[getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getDatabaseID\(\)](#), [getItemKey\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

[getType\(\)](#)

`public int getType()`

Overrides:

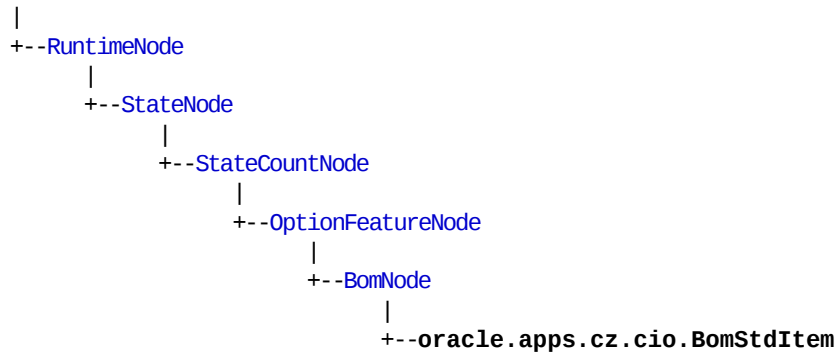
[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio BomStdItem

Syntax

public class BomStdItem extends [BomNode](#)

java.lang.Object



All Implemented Interfaces:

[IAtp](#), [IBomItem](#), [ICount](#), [IOption](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents configurable BOM models.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [BomNode](#)

[addAtpNotification\(String\)](#), [areOptionsCounted\(\)](#), [calculateAtpDate\(\)](#), [clearAtpDate\(\)](#), [clearAtpNotifications\(\)](#), [deselect\(\)](#), [getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getComponentCode\(\)](#), [getDefaultQuantity\(\)](#), [getDiscountedPrice\(\)](#), [getInventoryItemId\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getMaxQuantity\(\)](#), [getMaxSelected\(\)](#), [getMinQuantity\(\)](#), [getMinSelected\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [getState\(\)](#), [getUomCode\(\)](#), [hasDefaultQuantity\(\)](#), [hasMaxQuantity\(\)](#), [hasMaxSelected\(\)](#), [hasMinQuantity\(\)](#), [hasMinSelected\(\)](#), [isOptionMutexed\(\)](#), [isRequired\(\)](#), [isSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(\)](#), [select\(IOption\)](#), [setAtpDate\(Date\)](#), [setState\(int\)](#)

Methods inherited from class [OptionFeatureNode](#)

[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getExtendedPrice\(\)](#), [getPricingNotifications\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildById\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseId\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeId\(\)](#), [getSelectionLineId\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineId\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals\(\)](#), [getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(\)](#), [wait\(\)](#)

Methods inherited from interface [IBomItem](#)

[getComponentCode\(\)](#), [getInventoryItemId\(\)](#), [getMaxQuantity\(\)](#), [getMinQuantity\(\)](#), [getOrganizationId\(\)](#), [getPrimaryUomCode\(\)](#), [hasMaxQuantity\(\)](#), [hasMinQuantity\(\)](#), [isRequired\(\)](#)

Methods inherited from interface [IOptionFeature](#)

[areOptionsCounted\(\)](#), [deselect\(IOption\)](#), [getMaxSelected\(\)](#), [getMinSelected\(\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [hasMaxSelected\(\)](#), [hasMinSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from interface [IState](#)

Inherited Member Summary

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IOption](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IAtp](#)

[getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getDatabaseID\(\)](#), [getItemKey\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio

BooleanFeature

Syntax

```
public class BooleanFeature extends StateNode

java.lang.Object
|
+--RuntimeNode
    |
    +--StateNode
        |
        +--oracle.apps.cz.cio.BooleanFeature
```

All Implemented Interfaces:

[IRuntimeNode](#), [IState](#)

Description

Represents a feature with a boolean value.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [StateNode](#)

Inherited Member Summary

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [setState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio

CIO

Syntax

```
public class CIO extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.cio.CIO
```

Description

Implements a configuration integration object that can be used to create, save, restore and delete configurations.

Member Summary

Constructors

[CIO\(\)](#) Constructs a newly allocated configuration integration object.

Methods

clearLogicFile(Object)	Clears only the key specified file from the LCE file cache
clearLogicFileCache()	Clears all LogicFile objects from the logic file cache.
close()	Closes the CIO object and all associated runtime objects.
closeConfiguration(Configuration)	Deletes all runtime structure and memory associated with a configuration.
closeTraceFile()	
createConfiguration(int, Context)	Creates a new configuration based on a root model node ID representing a configurable product or component.
createConfiguration(int, int, Context)	
createConfiguration(int, int, Date, Context)	Creates a new BOM explosion configuration based on inventoryItemId, organizationId, and explosionDate representing a configurable product or component.
createConfiguration(String, Context)	Creates a new configuration based on a project name representing a configurable product or component.
getActiveModelPath()	Gets the current active model path.

Member Summary

[initTraceFile\(String\)](#)

[openTraceFile\(String\)](#)

[restoreConfiguration\(DbConfigHeader, Context\)](#) Restores a configuration from the database.

[restoreConfiguration\(int, int, Context\)](#) Restores a configuration from the database.

[setActiveModelPath\(String\)](#) Sets the path to the directory where the CIO will look for logic files, and where it will store logic files when generating them out of the database.

[trace\(String\)](#)

[tracing\(\)](#)

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Constructors

CIO()

```
public CIO()
```

Constructs a newly allocated configuration integration object.

Methods

clearLogicFile(Object)

```
public void clearLogicFile(java.lang.Object key)
```

Clears only the key specified file from the LCE file cache

clearLogicFileCache()

```
public void clearLogicFileCache()
```

Clears all LogicFile objects from the logic file cache.

close()

```
public void close()
```

Closes the CIO object and all associated runtime objects.

closeConfiguration(Configuration)

```
public void closeConfiguration(Configuration config)
```

Deletes all runtime structure and memory associated with a configuration.

Parameters:

`config` - the configuration to be deleted.

See Also:

[Configuration](#)

closeTraceFile()

```
public static void closeTraceFile()
```

createConfiguration(int, Context)

```
public Configuration createConfiguration(int rootNodeID,  
oracle.apps.fnd.common.Context ctx)
```

Creates a new configuration based on a root model node ID representing a configurable product or component.

Parameters:

`rootNodeID` - the ID of the DIO model node representing the product or configuration to be configured.

`ctx` - the Context object representing the application context

Returns:

a new configuration.

Throws:

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

[MissingFileException](#) - if a logic file cannot be found in the active model path

See Also:[Configuration](#)**createConfiguration(int, int, Context)**

```
public Configuration createConfiguration(int projectID, int rootNodeID,  
oracle.apps.fnd.common.Context ctx)
```

Deprecated.

Creates a new configuration based on a project ID and root model node ID both representing a configurable product or component.

Parameters:

`projectID` - the ID of the DIO project representing the product or configuration to be configured.

`rootNodeID` - the ID of the DIO model node representing the product or configuration to be configured.

`ctx` - the Context object representing the application context

Returns:

a new configuration.

Throws:

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

[MissingFileException](#) - if a logic file cannot be found in the active model path

See Also:[Configuration](#)**createConfiguration(int, int, Date, Context)**

```
public Configuration createConfiguration(int inventoryItemId, int  
organizationId, java.util.Date explosionDate, oracle.apps.fnd.common.Context  
ctx)
```

Creates a new BOM explosion configuration based on `inventoryItemId`, `organizationId`, and `explosionDate` representing a configurable product or component.

Parameters:

`inventoryItemId` - the inventory item id of the BOM explosion model

`organizationId` - the organization id of the BOM explosion model

`explosionDate` - the effective date of the BOM explosion model

`ctx` - the Context object representing the application context

Returns:

a new configuration.

Throws:

[NotOneProductException](#) - if the specified project contains more than one or no products

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

See Also:

[Configuration](#)

createConfiguration(String, Context)

```
public Configuration createConfiguration(java.lang.String projectName,  
oracle.apps.fnd.common.Context ctx)
```

Creates a new configuration based on a project name representing a configurable product or component.

Parameters:

`projectName` - the name of the DIO project representing the product or configuration to be configured.

`ctx` - the Context object representing the application context

Returns:

a new configuration.

Throws:

[NotOneProductException](#) - if the specified project contains more than one or no products

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

[MissingFileException](#) - if a logic file cannot be found in the active model path

See Also:

[Configuration](#)

getActiveModelPath()

```
public java.lang.String getActiveModelPath()
```

Gets the current active model path.

Returns:

the active model path.

initTraceFile(String)

```
public static void initTraceFile(java.lang.String filename)
```

openTraceFile(String)

```
public static void openTraceFile(java.lang.String filename)
```

restoreConfiguration(DbConfigHeader, Context)

```
public Configuration
```

```
restoreConfiguration(oracle.apps.cz.dio.config.DbConfigHeader header,
```

```
oracle.apps.fnd.common.Context ctx)
```

Restores a configuration from the database.

Parameters:

the - header containing information identifying the configuration to be restored.

ctx - the Context object representing the application context

Returns:

the restored configuration.

Throws:

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

[MissingFileException](#) - if a logic file cannot be found in the active model path

See Also:

[Configuration](#)

restoreConfiguration(int, int, Context)

```
public Configuration restoreConfiguration(int configHeaderID, int revNumber,  
oracle.apps.fnd.common.Context ctx)
```

Restores a configuration from the database.

Parameters:

the - ID of the header containing information identifying the configuration to be restored.

the - revision number of the header containing information identifying the configuration to be restored.

ctx - the Context object representing the application context

Returns:

the restored configuration.

Throws:

[LogicalException](#) - if a logic failure is encountered when initializing the configuration.

[MissingFileException](#) - if a logic file cannot be found in the active model path

See Also:

[Configuration](#)

setActiveModelPath(String)

```
public void setActiveModelPath(java.lang.String path)
```

Sets the path to the directory where the CIO will look for logic files, and where it will store logic files when generating them out of the database.

Parameters:

path - the path to the active model directory which should include the trailing path separator.

trace(String)

```
public static void trace(java.lang.String cmd)
```

tracing()

```
public static boolean tracing()
```

oracle.apps.cz.cio CompanionNode

Syntax

```
public class CompanionNode extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.cio.CompanionNode
```

Description

Provides methods for extracting values from a model based on property annotations.

Created by a parent [CompanionRoot](#) from a functional companion and used to get property-mapped feature information.

Member Summary

Methods

getBoolean(String)	Returns the value of a boolean feature.
getBoolean(String, boolean)	Returns the value of a boolean feature, or the default if features is not present.
getChildren()	Returns all of the CompanionNode children of this CompanionNode.
getDouble(String)	Returns the value of a double feature.
getDouble(String, double)	Returns the value of a double feature, or the default if features is not present.
getFeature(String)	Get the runtime node representing a particular feature based on its property annotation.
getInteger(String)	Returns the value of an integer feature.
getInteger(String, int)	Returns the value of an integer feature, or the default if features is not present.
getString(String)	Returns the value of a string feature.
getString(String, String)	Returns the value of a string feature, or the default if the feature is not present.
hasFeature(String)	Returns true if this CompanionNode contains the named feature.

Inherited Member Summary

Methods inherited from class `java.lang.Object`

`equals`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

Methods

getBoolean(String)

```
public boolean getBoolean(java.lang.String name)
```

Returns the value of a boolean feature.

getBoolean(String, boolean)

```
public boolean getBoolean(java.lang.String name, boolean dflt)
```

Returns the value of a boolean feature, or the default if features is not present.

getChildren()

```
public java.util.List getChildren()
```

Returns all of the `CompanionNode` children of this `CompanionNode`.

getDouble(String)

```
public double getDouble(java.lang.String name)
```

Returns the value of a double feature.

getDouble(String, double)

```
public double getDouble(java.lang.String name, double dflt)
```

Returns the value of a double feature, or the default if features is not present.

getFeature(String)

```
public IRuntimeNode getFeature(java.lang.String name)
```

Get the runtime node representing a particular feature based on its property annotation.

getInteger(String)

```
public int getInteger(java.lang.String name)
```

Returns the value of an integer feature.

getInteger(String, int)

```
public int getInteger(java.lang.String name, int dflt)
```

Returns the value of an integer feature, or the default if features is not present.

getString(String)

```
public java.lang.String getString(java.lang.String name)
```

Returns the value of a string feature.

getString(String, String)

```
public java.lang.String getString(java.lang.String name, java.lang.String dflt)
```

Returns the value of a string feature, or the default if the feature is not present.

hasFeature(String)

```
public boolean hasFeature(java.lang.String name)
```

Returns true if this CompanionNode contains the named feature.

oracle.apps.cz.cio CompanionRoot

Syntax

```
public abstract class CompanionRoot extends java.lang.Object
```

```
java.lang.Object
```

```
|
```

```
+--oracle.apps.cz.cio.CompanionRoot
```

Description

Provides functional companion implementors with property-based feature extraction.

An abstract class to be extended by functional companion implementors which attaches [CompanionNode](#) objects to runtime nodes and makes model features available to the function companion through a flexible system of property-based annotations.

In order to use this class, the implementor should provide implementations of [getNodeIdentifier\(\)](#), which returns the name of the property used to identify runtime nodes to which [CompanionNodes](#) will be attached, [getFeatureIdentifier\(\)](#) which returns the name of the property used to identify features of the [CompanionNode](#), and [getNodeClass\(String\)](#) which maps the value of the `getNodeIdentifier` property to the subclass of [CompanionNode](#) that should be instantiated to represent a particular node.

Member Summary

Constructors

CompanionRoot(IRuntimeNode)	Creates a tree of companion node objects based on property annotations.
---	---

Methods

getFeatureIdentifier()	Returns the name of the property used to identify companion features.
--	---

getNodeClass(String)	Maps a node type to the class used to represent the node.
--------------------------------------	---

Member Summary

getNodeIdentifier()	Returns the name of the property used to identify companion nodes.
getRootNodes()	Returns the root CompanionNode objects.

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

CompanionRoot(IRuntimeNode)

```
public CompanionRoot(IRuntimeNode root)
```

Creates a tree of companion node objects based on property annotations.

Methods

getFeatureIdentifier()

```
public abstract java.lang.String getFeatureIdentifier()
```

Returns the name of the property used to identify companion features.

getNodeClass(String)

```
public abstract java.lang.String getNodeClass(java.lang.String nodeType)
```

Maps a node type to the class used to represent the node.

getNodeIdentifier()

```
public abstract java.lang.String getNodeIdentifier()
```

Returns the name of the property used to identify companion nodes. Runtime nodes that have a property of this name will be mapped to CompanionNode objects. The value of the property will be mapped through the [getNodeClass\(String\)](#) method to determine which subclass of CompanionNode to instantiate.

getRootNodes()

```
public java.util.List getRootNodes()
```

Returns the root CompanionNode objects.

oracle.apps.cz.cio CompanionValidationFailure

Syntax

```
public class CompanionValidationFailure extends ValidationFailure

java.lang.Object
|
+--StatusInfo
    |
    +--ValidationFailure
        |
        +--oracle.apps.cz.cio.CompanionValidationFailure
```

Description

Failure produced by a functional companion object.

Member Summary

Constructors

[CompanionValidationFailure\(String,
IRuntimeNode,
IFunctionalCompanion\)](#)

Methods

- [equals\(Object\)](#)
 - [getCompanion\(\)](#) Returns the companion in which this validation failure occurred.
 - [hashCode\(\)](#)
-

Inherited Member Summary

Fields inherited from class [ValidationFailure](#)

[COMPANION_FAILURE](#), [MAX_FAILURE](#), [MIN_FAILURE](#), [MIN0_FAILURE](#), [MINMAX_FAILURE](#), [RESOURCE_FAILURE](#), [RESTORE_FAILURE](#)

Inherited Member Summary

Fields inherited from class [StatusInfo](#)

[STATUS_DELETED](#), [STATUS_EXISTING](#), [STATUS_NEW](#)

Methods inherited from class [ValidationFailure](#)

[getMessage\(\)](#), [getMessage\(String\)](#), [getType\(\)](#), [toString\(\)](#)

Methods inherited from class [StatusInfo](#)

[getNode\(\)](#), [getStatus\(\)](#), [statusToString\(int\)](#), [toString\(boolean\)](#)

Methods inherited from class java.lang.Object

[getClass](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Constructors

CompanionValidationFailure(String, IRuntimeNode, IFunctionalCompanion)

```
public CompanionValidationFailure(java.lang.String message, IRuntimeNode node,  
IFunctionalCompanion companion)
```

Methods

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Overrides:

[equals\(Object\)](#) in class [ValidationFailure](#)

getCompanion()

```
public IFunctionalCompanion getCompanion()  
Returns the companion in which this validation failure occurred.
```

hashCode()

```
public int hashCode()
```

Overrides:

[hashCode\(\)](#) in class [StatusInfo](#)

oracle.apps.cz.cio Component

Syntax

```
public class Component extends ComponentNode

java.lang.Object
|
+--RuntimeNode
    |
    +--ComponentNode
        |
        +--oracle.apps.cz.cio.Component
```

All Implemented Interfaces:

[IIntegerMinMax](#), [IRuntimeNode](#)

Description

Implements the runtime configuration behavior of products and components.

Member Summary

Methods

getChildren()	Returns a list of this node's children.
getCount()	Returns the count of the associated component set.
getFuncCompByID(int)	Returns a particular functional companion based on its ID, null if no match.
getFuncCompByName(String)	Returns a particular functional companion based on its name, null if no match.
getFunctionalCompanions()	Returns a list of all functional companions associated with this component.
getInstanceNumber()	Returns the instance number of this component (1 if not in a component set).
getMax()	Returns the maximum of the design-time component.
getMin()	Returns the minimum of the design-time component.
getName()	Returns the name of this runtime node.
getType()	Returns the type of this runtime node.

Member Summary

hasMax()	Returns true if the design-time component has a maximum.
hasMin()	Returns true if the design-time component has a minimum.
instanceTypeToString(int)	
isRoot()	Returns true if this is the root component in the runtime tree.
isVirtual()	Returns true if this component is a virtual component.
lookupNodeID(Object)	Returns the Runtime Node in this component given the id
setName(String)	Sets the name of this component.

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [ComponentNode](#)

[getChildrenByType\(int\)](#), [isActive\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getChildren()

public java.util.List getChildren()
Returns a list of this node's children.

Overrides:

[getChildren\(\)](#) in class [RuntimeNode](#)

getCount()

public int getCount()
Returns the count of the associated component set.

getFuncCompByID(int)

public [IFunctionalCompanion](#) getFuncCompByID(int id)
Returns a particular functional companion based on its ID, null if no match.

getFuncCompByName(String)

public [IFunctionalCompanion](#) getFuncCompByName(java.lang.String name)
Returns a particular functional companion based on its name, null if no match.

getFunctionalCompanions()

public java.util.List getFunctionalCompanions()
Returns a list of all functional companions associated with this component.

getInstanceNumber()

public int getInstanceNumber()
Returns the instance number of this component (1 if not in a component set).

getMax()

public int getMax()
Returns the maximum of the design-time component.

getMin()

public int getMin()
Returns the minimum of the design-time component.

getName()

public java.lang.String getName()
Returns the name of this runtime node.

Overrides:

[getName\(\)](#) in class [RuntimeNode](#)

getType()

public int getType()
Returns the type of this runtime node.

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

hasMax()

public boolean hasMax()
Returns true if the design-time component has a maximum.

hasMin()

public boolean hasMin()
Returns true if the design-time component has a minimum.

instanceTypeToString(int)

public static java.lang.String instanceTypeToString(int instanceType)

isRoot()

public boolean isRoot()
Returns true if this is the root component in the runtime tree.

isVirtual()

public boolean isVirtual()
Returns true if this component is a virtual component.

lookupNodeID(Object)

public [IRuntimeNode](#) lookupNodeID(java.lang.Object id)
Returns the Runtime Node in this component given the id

Overrides:

[lookupNodeID\(Object\)](#) in class [RuntimeNode](#)

setName(String)

```
public void setName(java.lang.String newName)
```

Sets the name of this component. NOTE: The method setName() shouldn't be used and may be removed in a future release.

oracle.apps.cz.cio

ComponentNode

Syntax

public abstract class ComponentNode extends [RuntimeNode](#) implements [IIntegerMinMax](#)



Direct Known Subclasses:

[Component](#), [ComponentSet](#)

All Implemented Interfaces:

[IIntegerMinMax](#), [IRuntimeNode](#)

Description

Represents a set of configurable components.

Member Summary

Methods

getChildrenByType(int)	Returns a list of all children of a given type.
isActive()	Returns true if this node has been activated.

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Inherited Member Summary

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IIntegerMinMax](#)

[getMax\(\)](#), [getMin\(\)](#), [hasMax\(\)](#), [hasMin\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getChildrenByType(int)

```
public java.util.List getChildrenByType(int type)
```

Returns a list of all children of a given type.

Specified By:

[getChildrenByType\(int\)](#) in interface [IRuntimeNode](#)

Overrides:

[getChildrenByType\(int\)](#) in class [RuntimeNode](#)

isActive()

```
public boolean isActive()
```

Returns true if this node has been activated.

oracle.apps.cz.cio ComponentSet

Syntax

public class ComponentSet extends [ComponentNode](#)

java.lang.Object

|

+--[RuntimeNode](#)

|

+--[ComponentNode](#)

|

+--**oracle.apps.cz.cio.ComponentSet**

All Implemented Interfaces:

[IIntegerMinMax](#), [IRuntimeNode](#)

Description

Represents a set of configurable components.

Member Summary

Methods

[add\(\)](#)

[addConfigEventListener\(ICompSetEventListener\)](#) Add a listener that is notified when a component is added or deleted.

[delete\(Component\)](#)

[getChildByInstanceNumber\(int\)](#)

[getCount\(\)](#)

[getMax\(\)](#)

[getMin\(\)](#)

[getType\(\)](#)

[hasMax\(\)](#)

Member Summary

[hasMin\(\)](#)

[removeConfigEventListener\(ICompSetEventListener\)](#) Remove a listener that is notified when a component is added or deleted.

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [ComponentNode](#)

[getChildrenByType\(int\)](#), [isActive\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

[add\(\)](#)

```
public Component add()
```

[addConfigEventListener\(ICompSetEventListener\)](#)

```
public void addConfigEventListener(ICompSetEventListener listener)
```

Add a listener that is notified when a component is added or deleted.

delete(Component)

```
public void delete(Component component)
```

getChildByInstanceNumber(int)

```
public Component getChildByInstanceNumber(int instNum)
```

getCount()

```
public int getCount()
```

getMax()

```
public int getMax()
```

getMin()

```
public int getMin()
```

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

hasMax()

```
public boolean hasMax()
```

hasMin()

```
public boolean hasMin()
```

removeConfigEventListener(ICompSetEventListener)

```
public void removeConfigEventListener(ICompSetEventListener listener)  
Remove a listener that is notified when a component is added or deleted.
```

oracle.apps.cz.cio ConfigTransaction

Syntax

```
public class ConfigTransaction extends oracle.apps.cz.cio.BasicConfigAction

java.lang.Object
|
+--oracle.apps.cz.cio.BasicConfigAction
|
+--oracle.apps.cz.cio.ConfigTransaction
```

Description

Represents a configuration transaction.

Inherited Member Summary
Methods inherited from class oracle.apps.cz.cio.BasicConfigAction
setTracing
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

oracle.apps.cz.cio Configuration

Syntax

```
public class Configuration
```

```
oracle.apps.cz.cio.Configuration
```

Description

The top-level entry point to a configuration.

Member Summary

Fields

ATP_APPS_107_110	Represents Oracle Apps 10.7/11.0 ATP mode.
ATP_CALLBACK	Represents callback ATP mode.
ATP_DISABLED	For the case when no valid ATP parameter combination is found.
PRICE_APPS_107_110	Represents Oracle Apps 10.7/11.0 pricing mode.
PRICE_MULT_ITEMS	Represents multiple item callback pricing mode.
PRICE_SINGLE_ITEM	Represents single item callback pricing mode.
PRICING_DISABLED	For the case when no valid pricing parameter combination is found.

Methods

addConfigEventListener(IConfigEventListener)	Add a listener that is notified when a component is added or deleted.
addConfigMessage(String, String)	Adds a configuration message to be saved to the cz_config_messages table.
beginConfigTransaction()	Creates a new transaction.
beginConfigTransaction(boolean)	Creates a new transaction and specifies the auto commit mode.
calculateAtpDates()	Calculates ATP dates for all IAtp nodes in the tree.
calculateListPrices()	Calculates list prices of all IPrice nodes within configuration.
canPerform()	Returns true if there is at least one undone or not committed transaction that can be performed.

Member Summary

canUndo()	Returns true if there are performed transactions that can be undone.
clearConfigMessages()	Removes all configuration messages added by addConfigMessage.
close()	Close the Configuration object and all associated runtime objects.
commitConfigTransaction(ConfigTransaction)	Commits the given transaction if it matches with current one in the configuration.
endConfigTransaction(ConfigTransaction)	Ends the given transaction if it matches with current one in the configuration.
endDeltaList()	
finalizeWorkaround()	
getAltPricingAtpContext()	Returns context that was added to the configuration through setAltPricingAtpContext, or null if no alternate pricing/ATP context exists.
getAtpMode()	Will return the ATP mode that will be used by the configuration (ATP_APPS_107_110 or ATP_CALLBACK).
getCIO()	Gets the CIO that created this configuration.
getConfigHeaderCheckoutUser()	Gets the user who has the config header checked out.
getConfigHeaderDateCreated()	Gets the date when the config header was created.
getConfigHeaderDescription()	Gets the description of the config header.
getConfigHeaderEffectiveFrom()	Gets the date from which the config header is effective.
getConfigHeaderEffectiveTo()	Gets the date to which the config header is effective.
getConfigHeaderId()	Gets the id of the config header.
getConfigHeaderLastUpdateDate()	Gets the date when the config header was last updated.
getConfigHeaderName()	Gets the name of the config header.
getConfigHeaderNote()	Gets the note of the config header.
getConfigHeaderNumberQuotesUsedIn()	Gets the config header number quotes used in.
getConfigHeaderOpportunityHeaderId()	Gets the opportunity header id of the config header .
getConfigHeaderRevision()	Gets the revision of the config header.
getConfigHeaderStatus()	Gets the status of the config header.

Member Summary

getConfigHeaderUiDefinitionId()	Gets the UI definition id in the config header.
getConfigHeaderUserIdCreated()	Gets the id of the user who created the config header.
getContext()	Returns the Context object associated with this configuration
getInitParameters()	Gets initialization parameters, which are stored in a NameValuePairSet object.
getLastContradiction()	Returns the most recent contradiction.
getLogicConfig()	
getNodeFromPath(String)	
getPathForNode(IRuntimeNode)	
getPricingMode()	Will return the pricing mode that will be used by the configuration (PRICE_APPS_107_110, PRICE_MULT_ITEMS, or PRICE_SINGLE_ITEM).
getProjectID()	Gets the database ID of the project from which this configuration was created.
getRootBomModel()	Returns the root BOM Model node, if there is associated with this configuration.
getRootBomModel(int, int)	Returns the root BOM Model node with the given inventory item ID and organization ID
getRootComponent()	Gets the root product or component of the configuration.
getRootComponentDbId()	Gets the database id of the root component.
getRuntimeNode(int)	Fetches a runtime node based on a runtime ID.
getSelectedItems()	Gets a collection of status info objects describing all selected items in the configuration.
getStringDeltaList()	
getTotalDiscountedPrice()	Returns rolled up discounted price of the configuration.
getTransactionDepth()	Returns the number of nested transactions (depth).
getUnsatisfiedItems()	Gets a collection of status info objects describing all unsatisfied items in the configuration.
getValidationFailures()	Gets a collection of validation failures describing current problems with the configuration.
isUnsatisfied()	Returns <code>true</code> if the configuration is incomplete.
perform()	Perform the next pending transaction.

Member Summary

<code>removeConfigEventListener(IConfigEventListener)</code>	Remove a listener that is notified when a component is added or deleted.
<code>restartConfiguration(boolean)</code>	Restart the current configuration instance without destroying the objects NOTE: Currently, it works only for values and states.
<code>rollbackConfigTransaction(ConfigTransaction)</code>	Rolls back the given transaction if it matches with current one in the configuration.
<code>save()</code>	Saves the restored configuration.
<code>saveAs(int, int)</code>	Saves over the current configuration.
<code>saveNew()</code>	Saves a whole new configuration.
<code>saveNewRev()</code>	Saves a new revision of the restored configuration.
<code>setAltPricingAtpContext(Context)</code>	If pricing and ATP information should be retrieved from an alternate Apps database, the <code>setAltPricingAtpContext</code> method should be called immediately after the Configuration is created.
<code>setConfigHeaderCheckoutUser(String)</code>	Sets the user who has checked out the config header.
<code>setConfigHeaderDateCreated(Timestamp)</code>	Sets the config header creation date.
<code>setConfigHeaderDescription(String)</code>	Sets the config header description.
<code>setConfigHeaderEffectiveFrom(Timestamp)</code>	Sets the date from which the config header is effective.
<code>setConfigHeaderEffectiveTo(Timestamp)</code>	Sets the date to which the config header is effective.
<code>setConfigHeaderName(String)</code>	Sets the config header name.
<code>setConfigHeaderNote(String)</code>	Sets the config header note.
<code>setConfigHeaderOpportunityHeaderId(int)</code>	Sets the config header opportunity header id.
<code>setConfigHeaderUiDefinitionId(int)</code>	Sets the config header UI definition id.
<code>setInitParameters(NameValuePairSet)</code>	Sets configuration initialization parameters, e.g.
<code>startDeltaList()</code>	
<code>undo()</code>	Undo the previous transaction.

Fields

ATP_APPS_107_110

```
public static final int ATP_APPS_107_110
```

Represents Oracle Apps 10.7/11.0 ATP mode.

ATP_CALLBACK

```
public static final int ATP_CALLBACK
```

Represents callback ATP mode.

ATP_DISABLED

```
public static final int ATP_DISABLED
```

For the case when no valid ATP parameter combination is found.

PRICE_APPS_107_110

```
public static final int PRICE_APPS_107_110
```

Represents Oracle Apps 10.7/11.0 pricing mode.

PRICE_MULT_ITEMS

```
public static final int PRICE_MULT_ITEMS
```

Represents multiple item callback pricing mode.

PRICE_SINGLE_ITEM

```
public static final int PRICE_SINGLE_ITEM
```

Represents single item callback pricing mode.

PRICING_DISABLED

```
public static final int PRICING_DISABLED
```

For the case when no valid pricing parameter combination is found.

Methods

addConfigEventListener(IConfigEventListener)

```
public void addConfigEventListener(IConfigEventListener listener)
```

Add a listener that is notified when a component is added or deleted.

addConfigMessage(String, String)

```
public void addConfigMessage(java.lang.String keyword, java.lang.String message)
```

Adds a configuration message to be saved to the cz_config_messages table.

Messages are cleared from the Configuration object when the configuration and messages are saved.

Parameters:

keyword - keyword describing the type of message, e.g. "CONTRADICTION"

message - message string

beginConfigTransaction()

```
public ConfigTransaction beginConfigTransaction()
```

Creates a new transaction.

Returns:

a reference to the newly created transaction.

beginConfigTransaction(boolean)

```
public ConfigTransaction beginConfigTransaction(boolean autoCommit)
```

Creates a new transaction and specifies the auto commit mode.

Returns:

a reference to the newly created transaction.

calculateAtpDates()

```
public java.util.Date calculateAtpDates()
```

Calculates ATP dates for all IAtp nodes in the tree. ATP values can then be retrieved using IAtp.getAtpDate().

Returns:

configuration level ATP date if calculated, null if not

Throws:

[AtpUnavailableException](#) - thrown if configuration initialization parameters required to run ATP check have not all been provided

calculateListPrices()

```
public void calculateListPrices()
```

Calculates list prices of all IPrice nodes within configuration. Prices are retrieved through IPrice.getListPrice.

canPerform()

```
public boolean canPerform()
```

Returns true if there is at least one undone or not committed transaction that can be performed.

canUndo()

```
public boolean canUndo()
```

Returns true if there are performed transactions that can be undone.

clearConfigMessages()

```
public void clearConfigMessages()
```

Removes all configuration messages added by addConfigMessage.

close()

```
public void close()
```

Close the Configuration object and all associated runtime objects.

commitConfigTransaction(ConfigTransaction)

```
public void commitConfigTransaction(ConfigTransaction transaction)
```

Commits the given transaction if it matches with current one in the configuration.

Parameters:

a - transaction reference.

endConfigTransaction(ConfigTransaction)

```
public void endConfigTransaction(ConfigTransaction transaction)
```

Ends the given transaction if it matches with current one in the configuration.

Parameters:

a - transaction reference.

endDeltaList()

```
public void endDeltaList()
```

finalizeWorkaround()

```
public void finalizeWorkaround()
```

getAltPricingAtpContext()

```
public oracle.apps.fnd.common.Context getAltPricingAtpContext()
```

Returns context that was added to the configuration through `setAltPricingAtpContext`, or null if no alternate pricing/ATP context exists.

getAtpMode()

```
public int getAtpMode()
```

Will return the ATP mode that will be used by the configuration (ATP_APPS_107_110 or ATP_CALLBACK).

Throws:

[AtpUnavailableException](#) - thrown if a complete set of ATP parameters has not been provided through `setInitParameters`.

getCIO()

```
public CIO getCIO()
```

Gets the CIO that created this configuration.

Returns:

the CIO that created this configuration.

See Also:

[CIO](#)

getConfigHeaderCheckoutUser()

```
public java.lang.String getConfigHeaderCheckoutUser()
```

Gets the user who has the config header checked out.

Returns:

the config header checkout user.

getConfigHeaderDateCreated()

```
public java.sql.Timestamp getConfigHeaderDateCreated()
```

Gets the date when the config header was created.

Returns:

the config header creation date.

getConfigHeaderDescription()

```
public java.lang.String getConfigHeaderDescription()
```

Gets the description of the config header.

Returns:

the config header description.

getConfigHeaderEffectiveFrom()

```
public java.sql.Timestamp getConfigHeaderEffectiveFrom()
```

Gets the date from which the config header is effective.

Returns:

the config header 'Effective From' date.

getConfigHeaderEffectiveTo()

```
public java.sql.Timestamp getConfigHeaderEffectiveTo()
```

Gets the date to which the config header is effective.

Returns:

the config header 'Effective To' date.

getConfigHeaderId()

```
public int getConfigHeaderId()
```

Gets the id of the config header.

Returns:

the config header id.

getConfigHeaderLastUpdateDate()

```
public java.sql.Timestamp getConfigHeaderLastUpdateDate()  
Gets the date when the config header was last updated.
```

Returns:

the config header last update date.

getConfigHeaderName()

```
public java.lang.String getConfigHeaderName()  
Gets the name of the config header.
```

Returns:

the config header name.

getConfigHeaderNote()

```
public java.lang.String getConfigHeaderNote()  
Gets the note of the config header.
```

Returns:

the config header note.

getConfigHeaderNumberQuotesUsedIn()

```
public int getConfigHeaderNumberQuotesUsedIn()  
Gets the config header number quotes used in.
```

Returns:

the config header number quotes used in.

getConfigHeaderOpportunityHeaderId()

```
public int getConfigHeaderOpportunityHeaderId()  
Gets the opportunity header id of the config header .
```

Returns:

the config header opportunity header id.

getConfigHeaderRevision()

```
public int getConfigHeaderRevision()
```

Gets the revision of the config header.

Returns:

the config header revision.

getConfigHeaderStatus()

```
public java.lang.String getConfigHeaderStatus()
```

Gets the status of the config header.

Returns:

the config header status.

getConfigHeaderUiDefinitionId()

```
public int getConfigHeaderUiDefinitionId()
```

Gets the UI definition id in the config header.

Returns:

the config header UI definition id.

getConfigHeaderUserIdCreated()

```
public int getConfigHeaderUserIdCreated()
```

Gets the id of the user who created the config header.

Returns:

the config header user id created.

getContext()

```
public oracle.apps.fnd.common.Context getContext()
```

Returns the Context object associated with this configuration

getInitParameters()

```
public NameValuePairSet getInitParameters()
```

Gets initialization parameters, which are stored in a NameValuePairSet object.

Returns:

initParameters object

getLastContradiction()

public [LogicalException](#) getLastContradiction()
Returns the most recent contradiction.

getLogicConfig()

public oracle.apps.cz.logic.LogicConfig getLogicConfig()

getNodeFromPath(String)

public [IRuntimeNode](#) getNodeFromPath(java.lang.String path)

getPathForNode(IRuntimeNode)

public java.lang.String getPathForNode([IRuntimeNode](#) node)

getPricingMode()

public int getPricingMode()
Will return the pricing mode that will be used by the configuration (PRICE_APPS_107_110, PRICE_MULT_ITEMS, or PRICE_SINGLE_ITEM).

Throws:

[PricingUnavailableException](#) - thrown if a complete set of pricing parameters has not been provided through setInitParameters.

getProjectID()

public int getProjectID()
Gets the database ID of the project from which this configuration was created.

Returns:

the project ID.

getRootBomModel()

public [BomModel](#) getRootBomModel()
Returns the root BOM Model node, if there is associated with this configuration.

getRootBomModel(int, int)

public [BomModel](#) getRootBomModel(int inventoryItemId, int organizationId)
Returns the root BOM Model node with the given inventory item ID and organization ID

getRootComponent()

public [Component](#) getRootComponent()
Gets the root product or component of the configuration.

Returns:

the root product or component.

See Also:

[Component](#)

getRootComponentDbId()

public int getRootComponentDbId()
Gets the database id of the root component.

Returns:

the root component db id.

getRuntimeNode(int)

public [IRuntimeNode](#) getRuntimeNode(int runtimeID)
Fetches a runtime node based on a runtime ID.

Parameters:

runtimeID - the runtime ID of the desired node.

Returns:

the corresponding runtime node.

See Also:

[IRuntimeNode](#)

getSelectedItems()

`public java.util.Collection getSelectedItems()`
Gets a collection of status info objects describing all selected items in the configuration.

Returns:

the collection of status info objects.

getStringDeltaList()

`public java.lang.String getStringDeltaList()`

getTotalDiscountedPrice()

`public double getTotalDiscountedPrice()`
Returns rolled up discounted price of the configuration. Discounted prices on selected items are available after this call through `IPrice.getDiscountedPrice`.

getTransactionDepth()

`public int getTransactionDepth()`
Returns the number of nested transactions (depth).

Returns:

the transaction depth.

getUnsatisfiedItems()

`public java.util.Collection getUnsatisfiedItems()`
Gets a collection of status info objects describing all unsatisfied items in the configuration.

Returns:

the collection of status info objects.

getValidationFailures()

`public java.util.Collection getValidationFailures()`
Gets a collection of validation failures describing current problems with the configuration.

Returns:

the collection of validation failures.

isUnsatisfied()

```
public boolean isUnsatisfied()
```

Returns `true` if the configuration is incomplete.

Returns:

a boolean indicating whether the configuration is unsatisfied.

perform()

```
public void perform()
```

Perform the next pending transaction.

removeConfigEventListener(IConfigEventListener)

```
public void removeConfigEventListener(IConfigEventListener listener)
```

Remove a listener that is notified when a component is added or deleted.

restartConfiguration(boolean)

```
public void restartConfiguration(boolean checkValidations)
```

Restart the current configuration instance without destroying the objects NOTE:

Currently, it works only for values and states. Additions and deletions are not being restarted. Therefore, after `restart()` you get the latest (before the restart) component instances and cannot undo an instance addition or deletion.

rollbackConfigTransaction(ConfigTransaction)

```
public void rollbackConfigTransaction(ConfigTransaction transaction)
```

Rolls back the given transaction if it matches with current one in the configuration.

Parameters:

a - transaction reference.

save()

```
public void save()
```

Saves the restored configuration.

Throws:

[NoConfigHeaderException](#) - when this configuration hasn't been previously saved. Consider calling `SaveNew()`.

`ConfigOverwriteNotAllowedException` - when this configuration is "read only". Consider calling `SaveNewRev()` or `SaveNew()`.

saveAs(int, int)

```
public void saveAs(int configHeaderID, int revNumber)
```

Saves over the current configuration. It uses the passed ID and revision number to open a `ConfigHeader` object and to save the configuration into it

Throws:

`ConfigOverwriteNotAllowedException` - when this configuration is "read only". Consider calling `SaveNewRev()` or `SaveNew()`.

saveNew()

```
public void saveNew()
```

Saves a whole new configuration.

saveNewRev()

```
public void saveNewRev()
```

Saves a new revision of the restored configuration.

Throws:

[NoConfigHeaderException](#) - when this configuration hasn't been previously saved. Consider calling `SaveNew()`.

setAltPricingAtpContext(Context)

```
public void setAltPricingAtpContext(oracle.apps.fnd.common.Context ctx)
```

If pricing and ATP information should be retrieved from an alternate Apps database, the `setAltPricingAtpContext` method should be called immediately after the `Configuration` is created.

Parameters:

`ctx` - Context which represents session on database from which pricing and ATP information is retrieved

setConfigHeaderCheckoutUser(String)

```
public void setConfigHeaderCheckoutUser(java.lang.String user)
```

Sets the user who has checked out the config header.

setConfigHeaderDateCreated(Timestamp)

```
public void setConfigHeaderDateCreated(java.sql.Timestamp dateCreated)
```

Sets the config header creation date.

setConfigHeaderDescription(String)

```
public void setConfigHeaderDescription(java.lang.String description)
```

Sets the config header description.

setConfigHeaderEffectiveFrom(Timestamp)

```
public void setConfigHeaderEffectiveFrom(java.sql.Timestamp effFrom)
```

Sets the date from which the config header is effective.

setConfigHeaderEffectiveTo(Timestamp)

```
public void setConfigHeaderEffectiveTo(java.sql.Timestamp effTo)
```

Sets the date to which the config header is effective.

setConfigHeaderName(String)

```
public void setConfigHeaderName(java.lang.String name)
```

Sets the config header name.

setConfigHeaderNote(String)

```
public void setConfigHeaderNote(java.lang.String note)
```

Sets the config header note.

setConfigHeaderOpportunityHeaderId(int)

```
public void setConfigHeaderOpportunityHeaderId(int id)
```

Sets the config header opportunity header id.

setConfigHeaderUiDefinitionId(int)

```
public void setConfigHeaderUiDefinitionId(int id)
```

Sets the config header UI definition id.

setInitParameters(NameValuePairSet)

```
public void setInitParameters(NameValuePairSet initParameters)
```

Sets configuration initialization parameters, e.g. order header information. All parameter values should be provided as String objects.

To use the callback pricing mechanism, the following parameters must be provided: "pricing_package_name" (required), "price_mult_items_proc" or "price_single_item_proc" (one is required), "configurator_session_key" (required)

To use Apps 10.7/11.0 pricing for BomNodes ("AMNT" pricing method only), the following parameters must be provided: "price_list_id" (required), "pricing_attribute1" (optional), "pricing_attribute2" (optional), "pricing_attribute3" (optional), "pricing_attribute4" (optional), "pricing_attribute5" (optional), "pricing_attribute6" (optional), "pricing_attribute7" (optional), "pricing_attribute8" (optional), "pricing_attribute9" (optional), "pricing_attribute10" (optional), "pricing_attribute11" (optional), "pricing_attribute12" (optional), "pricing_attribute13" (optional), "pricing_attribute14" (optional), "pricing_attribute15" (optional), "ship_to_site_use_id" (optional), "customer_id" (optional), "invoice_to_site_use_id" (optional), "po_number" (optional), "agreement_id" (optional), "agreement_type_code" (optional), "order_type_id" (optional), "gsa" (optional).

NOTE: If the callback parameters and price_list_id are both provided, then the pricing callback will be run to determine prices.

To use the callback ATP mechanism, the following parameters must be provided: "atp_package_name" (required), "get_atp_dates_proc" (required), "configurator_session_key" (required), "warehouse_id" (required), "requested_date" (optional), and either "ship_to_org_id" (required) or "customer_id" and "customer_site_id" (required),

To use Apps 10.7/11.0 ATP calculation methods, the following parameters must be provided: "user_id" (required), "application_id" (required), "responsibility_id" (required), "atp_timeout" (required)

startDeltaList()

```
public void startDeltaList()
```

undo()

```
public void undo()
```

Undo the previous transaction.

oracle.apps.cz.cio CountFeature

Syntax

public class CountFeature extends [StateCountNode](#) implements [IInteger](#),
[IIntegerMinMax](#)



All Implemented Interfaces:

[ICount](#), [IInteger](#), [IIntegerMinMax](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents a countable integer feature. A count feature is similar to an integer feature except that its minimum value must be greater than or equal to zero.

Member Summary

Methods

[getIntValue\(\)](#)
[getMax\(\)](#)
[getMin\(\)](#)
[getType\(\)](#)
[hasMax\(\)](#)
[hasMin\(\)](#)
[setIntValue\(int\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [setState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IInteger](#)

[isUnknown\(\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [setState\(int\)](#)

Methods inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods**getIntValue()**

```
public int getIntValue()
```

Specified By:

[getIntValue\(\)](#) in interface [IInteger](#)

getMax()

```
public int getMax()
```

Specified By:

[getMax\(\)](#) in interface [IIntegerMinMax](#)

getMin()

```
public int getMin()
```

Specified By:

[getMin\(\)](#) in interface [IIntegerMinMax](#)

getType()

```
public int getType()
```

Specified By:

[getType\(\)](#) in interface [IRuntimeNode](#)

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

hasMax()

```
public boolean hasMax()
```

Specified By:

[hasMax\(\)](#) in interface [IIntegerMinMax](#)

hasMin()

```
public boolean hasMin()
```

Specified By:

[hasMin\(\)](#) in interface [IIntegerMinMax](#)

setIntValue(int)

```
public void setIntValue(int newValue)
```

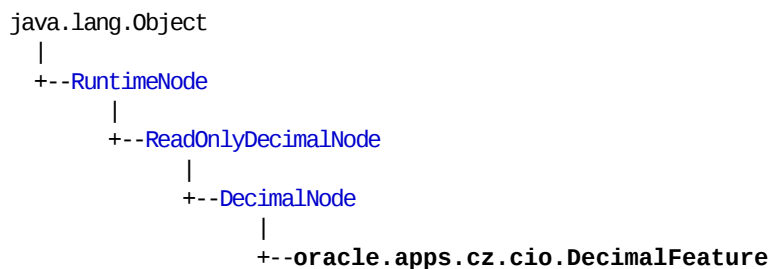
Specified By:

[setIntValue\(int\)](#) in interface [IInteger](#)

oracle.apps.cz.cio DecimalFeature

Syntax

public class DecimalFeature extends [DecimalNode](#) implements [IDecimalMinMax](#)



All Implemented Interfaces:

[IDecimal](#), [IDecimalMinMax](#), [IReadOnlyDecimal](#), [IRuntimeNode](#)

Description

Represents a feature with a decimal value.

Member Summary

Methods

[getMax\(\)](#)

[getMin\(\)](#)

[getType\(\)](#)

[hasMax\(\)](#)

[hasMin\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [DecimalNode](#)

[setDecimalValue\(double\)](#), [toString\(\)](#), [unset\(\)](#)

Methods inherited from class [ReadOnlyDecimalNode](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IDecimal](#)

[setDecimalValue\(double\)](#), [unset\(\)](#)

Methods inherited from interface [IReadingOnlyDecimal](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getMax()

```
public double getMax()
```

Specified By:[getMax\(\)](#) in interface [IDecimalMinMax](#)**getMin()**

```
public double getMin()
```

Specified By:[getMin\(\)](#) in interface [IDecimalMinMax](#)**getType()**

```
public int getType()
```

Specified By:[getType\(\)](#) in interface [IRuntimeNode](#)**Overrides:**[getType\(\)](#) in class [RuntimeNode](#)**hasMax()**

```
public boolean hasMax()
```

Specified By:[hasMax\(\)](#) in interface [IDecimalMinMax](#)**hasMin()**

```
public boolean hasMin()
```

Specified By:[hasMin\(\)](#) in interface [IDecimalMinMax](#)

oracle.apps.cz.cio DecimalNode

Syntax

public abstract class DecimalNode extends [ReadOnlyDecimalNode](#) implements [IDecimal](#)



Direct Known Subclasses:

[DecimalFeature](#), [Resource](#), [Total](#)

All Implemented Interfaces:

[IDecimal](#), [IReadOnlyDecimal](#), [IRuntimeNode](#)

Description

An abstract class implementing behavior common to objects with a decimal value.

Member Summary

Methods

[setDecimalValue\(double\)](#)

[toString\(\)](#)

[unset\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [ReadOnlyDecimalNode](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaselD\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IReadOnlyDecimal](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaselD\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

setDecimalValue(double)

```
public void setDecimalValue(double newValue)
```

Specified By:

[setDecimalValue\(double\)](#) in interface [IDecimal](#)

toString()

```
public java.lang.String toString()
```

Overrides:

[toString\(\)](#) in class [ReadOnlyDecimalNode](#)

unset()

public void unset()

Specified By:

[unset\(\)](#) in interface [IDecimal](#)

oracle.apps.cz.cio Factory

Syntax

public class Factory extends java.lang.Object



Description

Provides a class factory for the CIO to clients that do not support arguments to constructors.

Member Summary

Constructors

[Factory\(\)](#)

Methods

createCIO()	Creates an instance of the CIO.
createContext(String, String, String, String)	Creates the database context object which is required by most CIO methods.
createContext(String, String, String, String, String, String, String)	Creates the database context object which is required by most CIO methods.
loadDriver(String)	Loads the JDBC driver named by the argument.

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

Factory()

```
public Factory()
```

Methods

createCIO()

```
public CIO createCIO()  
Creates an instance of the CIO.
```

createContext(String, String, String, String)

```
public oracle.apps.fnd.common.Context createContext(java.lang.String url,  
java.lang.String uname, java.lang.String pwd, java.lang.String owner)  
Creates the database context object which is required by most CIO methods. This  
method requires a database user and password.
```

Parameters:

url - database connection URL that specifies JDBC driver and datasource

uname - database username

pwd - database password

owner - SellingPoint schema owner

createContext(String, String, String, String, String, String, String)

```
public oracle.apps.fnd.common.Context createContext(java.lang.String url,  
java.lang.String appsUsername, java.lang.String appsPassword, java.lang.String  
gatewayUsername, java.lang.String gatewayPassword, java.lang.String fndNam,  
java.lang.String applServerId)  
Creates the database context object which is required by most CIO methods. This  
method uses Apps FND authentication to validate the user.
```

Parameters:

url - database connection URL, identifies JDBC driver and data source

appsUsername - Oracle Applications user name

appsPassword - Oracle Applications password for appsUsername

gatewayUsername - gateway user name

gatewayPassword - password for gatewayUsername

fndNam - Apps schema owner

applServerId - application server ID, only used if its security feature is ON (OFF by default)

loadDriver(String)

```
public void loadDriver(java.lang.String driver)
```

Loads the JDBC driver named by the argument.

Throws:

`ClassNotFoundException` - if the driver cannot be loaded.

oracle.apps.cz.cio FuncCompCreationException

Syntax

```
public class FuncCompCreationException extends java.lang.RuntimeException

java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.lang.RuntimeException
            |
            +--oracle.apps.cz.cio.FuncCompCreationException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signalled if a functional companion cannot be created.

Member Summary	
Methods	
getDescription()	Returns the description of the failed companion.
getID()	Returns the database ID of the failed companion.
getName()	Returns the name of the failed companion.
getProgString()	Returns the program string used when trying to create the companion.

Inherited Member Summary	
Methods inherited from class java.lang.Throwable	
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString	

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods**getDescription()**

public java.lang.String getDescription()
Returns the description of the failed companion.

getID()

public int getID()
Returns the database ID of the failed companion.

getName()

public java.lang.String getName()
Returns the name of the failed companion.

getProgString()

public java.lang.String getProgString()
Returns the program string used when trying to create the companion.

oracle.apps.cz.cio FuncCompMessageException

Syntax

```
public class FuncCompMessageException extends java.lang.RuntimeException

java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--java.lang.RuntimeException
            |
            +--oracle.apps.cz.cio.FuncCompMessageException
```

All Implemented Interfaces:

java.io.Serializable

Description

FuncCompMessageException is designed to be thrown from a functional companion's autoConfigure() method code when the author of the companion wants to display a BENIGN message in a dialog box to the end user. Throwing this will cause the UI Server to commit the outermost transaction. To produce a fatal error message throw RuntimeException

Member Summary

Constructors

[FuncCompMessageException\(IFunctionalCompanion, String\)](#) Constructs the exception object.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

Inherited Member Summary

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors**FuncCompMessageException(IFunctionalCompanion, String)**

```
public FuncCompMessageException(IFunctionalCompanion fc, java.lang.String  
message)
```

Constructs the exception object.

Parameters:

fc - the FunctionalCompanion from which exception is thrown, not null

message - the message to be shown to the end user

oracle.apps.cz.cio FunctionalCompanion

Syntax

public class FunctionalCompanion extends java.lang.Object implements [IFunctionalCompanion](#)



Direct Known Subclasses:

[AutoFunctionalCompanion](#)

All Implemented Interfaces:

[IFunctionalCompanion](#)

Description

Base object on which user functional companions can be based.

Member Summary

Constructors

[FunctionalCompanion\(\)](#)

Methods

autoConfigure()	Does nothing.
generateOutput()	Does nothing.
generateOutput(HttpServletResponse)	Does nothing.
getDescription()	Returns the description of the functional companion.
getID()	Returns the database ID of the functional companion.
getName()	Returns the name of the functional companion.
getRuntimeNode()	Returns the runtime node to which this functional is associated.

Member Summary	
initialize(IRuntimeNode, String, String, int)	Saves the parameters in member variables.
terminate()	Does nothing.
toString()	
validate()	Does nothing.

Inherited Member Summary
Methods inherited from class java.lang.Object
<code>equals, getClass, hashCode, notify, notifyAll, wait, wait, wait</code>

Constructors

FunctionalCompanion()

```
public FunctionalCompanion()
```

Methods

autoConfigure()

```
public void autoConfigure()  
Does nothing.
```

Specified By:
[autoConfigure\(\)](#) in interface [IFunctionalCompanion](#)

generateOutput()

```
public java.lang.String generateOutput()  
Does nothing. Returns null.
```

Specified By:
[generateOutput\(\)](#) in interface [IFunctionalCompanion](#)

generateOutput(HttpServletResponse)

public void generateOutput(javax.servlet.http.HttpServletResponse response)
Does nothing.

Specified By:

[generateOutput\(HttpServletResponse\)](#) in interface [IFunctionalCompanion](#)

getDescription()

public java.lang.String getDescription()
Returns the description of the functional companion.

Specified By:

[getDescription\(\)](#) in interface [IFunctionalCompanion](#)

getID()

public int getID()
Returns the database ID of the functional companion.

Specified By:

[getID\(\)](#) in interface [IFunctionalCompanion](#)

getName()

public java.lang.String getName()
Returns the name of the functional companion.

Specified By:

[getName\(\)](#) in interface [IFunctionalCompanion](#)

getRuntimeNode()

public [IRuntimeNode](#) getRuntimeNode()
Returns the runtime node to which this functional is associated.

Specified By:

[getRuntimeNode\(\)](#) in interface [IFunctionalCompanion](#)

initialize(IRuntimeNode, String, String, int)

public void initialize([IRuntimeNode](#) node, java.lang.String name,
java.lang.String description, int id)
Saves the parameters in member variables.

Specified By:

[initialize\(IRuntimeNode, String, String, int\)](#) in interface [IFunctionalCompanion](#)

terminate()

public void terminate()
Does nothing.

Specified By:

[terminate\(\)](#) in interface [IFunctionalCompanion](#)

toString()

public java.lang.String toString()

Overrides:

java.lang.Object.toString() in class java.lang.Object

validate()

public java.util.List validate()
Does nothing.

Specified By:

[validate\(\)](#) in interface [IFunctionalCompanion](#)

oracle.apps.cz.cio FunctionalCompanionException

Syntax

```
public class FunctionalCompanionException
```

```
oracle.apps.cz.cio.FunctionalCompanionException
```

Description

This exception is used to indicate that an error occurred somewhere inside the functional companion.

Member Summary

Constructors

[FunctionalCompanionException\(Throwable\)](#) The message of the original exception will be the message of this exception.

[FunctionalCompanionException\(Throwable, String\)](#)

Constructors

FunctionalCompanionException(Throwable)

```
public FunctionalCompanionException(java.lang.Throwable ex)
```

The message of the original exception will be the message of this exception.

FunctionalCompanionException(Throwable, String)

```
public FunctionalCompanionException(java.lang.Throwable ex, java.lang.String message)
```

Parameters:

message - the message of the exception

oracle.apps.cz.cio

IAtp

Syntax

public interface IAtp extends [ICount](#)

All Known Subinterfaces:

[IBomItem](#)

All Superinterfaces:

[ICount](#)

Description

Implemented by objects that can have ATP calculated. This interface contains methods for getting available-to-promise (ATP) information, and methods to retrieve ATP errors/warnings/messages.

Member Summary

Methods

getAtpDate()	Retrieves last ATP date calculated by Configuration.calculateAtpDates for this item.
getAtpNotifications()	Returns string containing any ATP messages, warnings or errors generated for this node by the latest Configuration.calculateAtpDates call.
getDatabaseID()	Returns the database ID of the runtime node.
getItemKey()	Returns item key for items imported from Oracle Inventory / BOM.
getUomCode()	Returns unit of measure code for items imported from Oracle Inventory / BOM.

Inherited Member Summary

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods

getAtpDate()

```
public java.util.Date getAtpDate()
```

Retrieves last ATP date calculated by Configuration.calculateAtpDates for this item.

Returns:

ATP date

Throws:

[AtpUnavailableException](#) - thrown if ATP initialization parameters were not provided

[NoAtpCalculatedException](#) - thrown if ATP was never demanded or if the ATP procedure did not calculate an ATP date for this node

getAtpNotifications()

```
public java.lang.String getAtpNotifications()
```

Returns string containing any ATP messages, warnings or errors generated for this node by the latest Configuration.calculateAtpDates call.

getDatabaseID()

```
public int getDatabaseID()
```

Returns the database ID of the runtime node.

getItemKey()

```
public java.lang.String getItemKey()
```

Returns item key for items imported from Oracle Inventory / BOM. Item key is constructed from BOM_EXPLOSIONS field values: "[COMPONENT_CODE]:[EXPLOSION_TYPE]:[ORGANIZATION_ID]:[TOP_ITEM_ID]" Item key may be used by PL/SQL ATP procedures to calculate ATP for nodes. Returns null if node was not imported from Oracle Inventory / BOM.

getUomCode()

```
public java.lang.String getUomCode()
```

Returns unit of measure code for items imported from Oracle Inventory / BOM. The unit of measure may be used by PL/SQL pricing procedures to calculate ATP for nodes. Returns null if node was not imported from Oracle Inventory / BOM.

oracle.apps.cz.cio IBomItem

Syntax

public interface IBomItem extends [IOptionFeature](#), [IOption](#), [IPrice](#), [IAtp](#)

All Superinterfaces:

[IAtp](#), [ICount](#), [IOption](#), [IOptionFeature](#), [IPrice](#), [IState](#)

All Known Implementing Classes:

[BomNode](#)

Description

Implemented by all selectable BOM items.

Member Summary

Methods

getComponentCode()	Returns component code of item.
getInventoryItemId()	Returns Oracle Applications inventory_item_id of item.
getMaxQuantity()	Gets the maximum quantity.
getMinQuantity()	Gets the minimum quantity.
getOrganizationId()	Returns Oracle Applications organization_id of item.
getPrimaryUomCode()	Gets primary unit of measure code for item.
hasMaxQuantity()	Reuturns true if the BOM item has maximum quantity
hasMinQuantity()	Returns true if the BOM item has minimum quantity
isRequired()	Returns <code>true</code> if this is a required BOM item.

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Methods inherited from interface [IOptionFeature](#)

[areOptionsCounted\(\)](#), [deselect\(IOption\)](#), [getMaxSelected\(\)](#), [getMinSelected\(\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [hasMaxSelected\(\)](#), [hasMinSelected\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IOption](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IAtp](#)

[getAtpDate\(\)](#), [getAtpNotifications\(\)](#), [getDatabaseID\(\)](#), [getItemKey\(\)](#), [getUomCode\(\)](#)

Methods

getComponentCode()

```
public java.lang.String getComponentCode()
```

Returns component code of item. Component code is used to identify the item within an exploded bill of materials.

getInventoryItemId()

```
public int getInventoryItemId()
```

Returns Oracle Applications inventory_item_id of item.

getMaxQuantity()

```
public int getMaxQuantity()
```

Gets the maximum quantity.

getMinQuantity()

public int getMinQuantity()
Gets the minimum quantity.

getOrganizationId()

public int getOrganizationId()
Returns Oracle Applications organization_id of item.

getPrimaryUomCode()

public java.lang.String getPrimaryUomCode()
Gets primary unit of measure code for item.

hasMaxQuantity()

public boolean hasMaxQuantity()
Returns true if the BOM item has maximum quantity

hasMinQuantity()

public boolean hasMinQuantity()
Returns true if the BOM item has minimum quantity

isRequired()

public boolean isRequired()
Returns true if this is a required BOM item.

oracle.apps.cz.cio

ICompSetEventListener

Syntax

public interface ICompSetEventListener extends java.util.EventListener

All Superinterfaces:

java.util.EventListener

Description

Implemented by objects that want to find out about added components.

Member Summary

Methods

[notifyComponentAdded\(Component\)](#) Called when a component is added to the component set.

[notifyComponentDeleted\(Component \)](#) Called when a component is deleted from the component set.

Methods

notifyComponentAdded(Component)

public void notifyComponentAdded([Component](#) component)
Called when a component is added to the component set.

notifyComponentDeleted(Component)

public void notifyComponentDeleted([Component](#) component)
Called when a component is deleted from the component set.

oracle.apps.cz.cio IConfigEventListener

Syntax

```
public interface IConfigEventListener extends java.util.EventListener
```

All Superinterfaces:

java.util.EventListener

Description

Implemented by objects that want to find out about added components. This listener's methods are called as the result of user interaction, after a functional companion is initialized.

Member Summary

Methods

notifyComponentAdded(Component)	Called when a component is added to the configuration as the result of user interaction, after a functional companion is initialized.
notifyComponentDeleted(Component)	Called when a component is deleted from the configuration as the result of user interaction, after a functional companion is initialized.

Methods

notifyComponentAdded(Component)

```
public void notifyComponentAdded(Component component)
```

Called when a component is added to the configuration as the result of user interaction, after a functional companion is initialized.

notifyComponentDeleted(Component)

```
public void notifyComponentDeleted(Component component)
```

Called when a component is deleted from the configuration as the result of user interaction, after a functional companion is initialized.

oracle.apps.cz.cio
ICount

Syntax

public interface ICount

All Known Subinterfaces:

[IAtp](#), [IBomItem](#), [IOption](#), [IOptionFeature](#), [IPrice](#)

All Known Implementing Classes:

[StateCountNode](#)

Description

Implemented by objects that have an associated integer count.

Member Summary

Methods

getCount()	Gets the current count of this object.
setCount(int)	Sets the count of this object.
unset()	Retracts any user selection made toward this node

Methods

getCount()

public int getCount()
Gets the current count of this object.

Returns:

the current count of this object.

setCount(int)

public void setCount(int newCount)

Sets the count of this object.

unset()

```
public void unset()
```

Retracts any user selection made toward this node

oracle.apps.cz.cio

IDecimal

Syntax

public interface IDecimal extends [IReadOnlyDecimal](#)

All Superinterfaces:

[IReadOnlyDecimal](#)

All Known Implementing Classes:

[DecimalNode](#)

Description

Implemented by objects that can both get and set a decimal value.

Member Summary

Methods

setDecimalValue(double)	Sets the current value of this object.
unset()	Retracts any user selection made toward this node

Inherited Member Summary

Methods inherited from interface [IReadOnlyDecimal](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods

setDecimalValue(double)

public void setDecimalValue(double newValue)
Sets the current value of this object.

unset()

```
public void unset()
```

Retracts any user selection made toward this node

oracle.apps.cz.cio

IDecimalMinMax

Syntax

public interface IDecimalMinMax

All Known Implementing Classes:

[DecimalFeature](#)

Description

Implemented by objects that have a decimal minimum and maximum value.

Member Summary

Methods

getMax()	Get the maximum allowable value.
getMin()	Get the minimum allowable value.
hasMax()	Returns true if there is a maximum limit.
hasMin()	Returns true if there is a minimum limit.

Methods

getMax()

public double getMax()
Get the maximum allowable value.

getMin()

public double getMin()
Get the minimum allowable value.

hasMax()

public boolean hasMax()
Returns true if there is a maximum limit.

hasMin()

```
public boolean hasMin()  
Returns true if there is a minimum limit.
```

oracle.apps.cz.cio

IFunctionalCompanion

Syntax

public interface IFunctionalCompanion

All Known Implementing Classes:

[FunctionalCompanion](#)

Description

Implemented by functional companion objects attached to components in order to provide programatic functionality to a configuration model.

Member Summary

Fields	Functional Companion event types
FC_ON_AUTO_CONFIGURE	1004
FC_ON_GENERATE_OUTPUT	1006
FC_ON_NEW	1001
FC_ON_RESTORE	1002
FC_ON_SAVE	1003
FC_ON_VALIDATE	1005
Methods	
autoConfigure()	Performs a programmatic configuration step.
generateOutput()	Generates output for this component.
generateOutput(HttpServletResponse)	Generates output for this component.
getDescription()	Returns the description of the functional companion.
getID()	Returns the database ID of the functional companion.
getName()	Returns the name of the functional companion.
getRuntimeNode()	Returns the runtime node to which this component is attached.

Member Summary

<code>initialize(IRuntimeNode, String, String, int)</code>	Saves information about the model and performs any actions needed to initialize the companion.
<code>terminate()</code>	Performs any cleanup on this companion that needs to occur before the companion is destroyed.
<code>validate()</code>	Programatically checks that a configuration is valid and returns a list of ValidationFailure objects if there are failures, and null otherwise.

Fields

FC_ON_AUTO_CONFIGURE

`public static final int FC_ON_AUTO_CONFIGURE`
Functional Companion event type.

FC_ON_GENERATE_OUTPUT

`public static final int FC_ON_GENERATE_OUTPUT`
Functional Companion event type.

FC_ON_NEW

`public static final int FC_ON_NEW`
Functional Companion event type.

FC_ON_RESTORE

`public static final int FC_ON_RESTORE`
Functional Companion event type.

FC_ON_SAVE

`public static final int FC_ON_SAVE`
Functional Companion event type.

FC_ON_VALIDATE

`public static final int FC_ON_VALIDATE`
Functional Companion event type.

Methods

autoConfigure()

```
public void autoConfigure()
```

Performs a programmatic configuration step. Any modifications to the model should be performed here.

generateOutput()

```
public java.lang.String generateOutput()
```

Generates output for this component. This version is called in a thick client context where the user's machine can be addressed directly. Can modify the model, but this is not recommended practice.

generateOutput(HttpServletResponse)

```
public void generateOutput(javax.servlet.http.HttpServletResponse response)
```

Generates output for this component. This version is called in a thin client context where the user's browser is addressed indirectly by writing to the HttpServletResponse object. Can modify the model, but this is not recommended practice.

getDescription()

```
public java.lang.String getDescription()
```

Returns the description of the functional companion.

getID()

```
public int getID()
```

Returns the database ID of the functional companion.

getName()

```
public java.lang.String getName()
```

Returns the name of the functional companion.

getRuntimeNode()

```
public IRuntimeNode getRuntimeNode()
```

Returns the runtime node to which this component is attached.

initialize(IRuntimeNode, String, String, int)

```
public void initialize(IRuntimeNode node, java.lang.String name,  
java.lang.String description, int id)
```

Saves information about the model and performs any actions needed to initialize the companion. Should never attempt to modify the model.

terminate()

```
public void terminate()
```

Performs any cleanup on this companion that needs to occur before the companion is destroyed.

validate()

```
public java.util.List validate()
```

Programatically checks that a configuration is valid and returns a list of ValidationFailure objects if there are failures, and null otherwise. Should never attempt to modify the model.

oracle.apps.cz.cio

Integer

Syntax

public interface IInteger

All Known Implementing Classes:

[CountFeature](#), [IntegerNode](#)

Description

Implemented by objects that have an integer value.

Member Summary

Methods

getIntValue()	Get the current integer value of this object.
isUnknown()	Tells whether this feature is in unknown or known state.
setIntValue(int)	Set the current integer value of this object.
unset()	Retracts any user selection made toward this node

Methods

getIntValue()

public int getIntValue()
Get the current integer value of this object.

isUnknown()

public boolean isUnknown()
Tells whether this feature is in unknown or known state.

Returns:

true if the feature is unknown.

setIntValue(int)

`public void setIntValue(int newValue)`
Set the current integer value of this object.

unset()

`public void unset()`
Retracts any user selection made toward this node

oracle.apps.cz.cio

IntegerMinMax

Syntax

public interface IIntegerMinMax

All Known Implementing Classes:

[CountFeature](#), [IntegerFeature](#), [ComponentNode](#)

Description

Implemented by objects that have an integer minimum and maximum.

Member Summary

Methods

getMax()	Get the maximal allowable value for this object.
getMin()	Get the minimal allowable value for this object.
hasMax()	Returns true if there is a maximum limit.
hasMin()	Returns true if there is a minimum limit.

Methods

getMax()

public int getMax()
Get the maximal allowable value for this object.

getMin()

public int getMin()
Get the minimal allowable value for this object.

hasMax()

public boolean hasMax()
Returns true if there is a maximum limit.

hasMin()

```
public boolean hasMin()  
Returns true if there is a minimum limit.
```

oracle.apps.cz.cio IncompatibleInputException

Syntax

```
public class IncompatibleInputException extends java.lang.Exception

java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.IncompatibleInputException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signalled if a particular input is of different type than the node it is trying to restore over.

Member Summary

Methods

getInput()	Returns the input object where the mismatch occurred
getModelNode()	Returns the corresponding model node where the mismatch occurred

Inherited Member Summary

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods

getInput()

```
public oracle.apps.cz.dio.config.DbConfigInput getInput()
```

Returns the input object where the mismatch occurred

Returns:

the failed DbConfigInput object

getModelNode()

```
public oracle.apps.cz.dio.model.DbModelNode getModelNode()
```

Returns the corresponding model node where the mismatch occurred

Returns:

the failed DbConfigInput object

oracle.apps.cz.cio IntegerFeature

Syntax

public class IntegerFeature extends [IntegerNode](#) implements [IIntegerMinMax](#)



All Implemented Interfaces:

[IInteger](#), [IIntegerMinMax](#), [IRuntimeNode](#)

Description

Represents a feature with an integer value.

Member Summary

Methods

[getMax\(\)](#)

[getMin\(\)](#)

[getType\(\)](#)

[hasMax\(\)](#)

[hasMin\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [IntegerNode](#)

[getIntValue\(\)](#), [isUnknown\(\)](#), [setIntValue\(int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaselD\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IInteger](#)

[getIntValue\(\)](#), [isUnknown\(\)](#), [setIntValue\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaselD\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getMax()

```
public int getMax()
```

Specified By:

[getMax\(\)](#) in interface [IIntegerMinMax](#)

getMin()

```
public int getMin()
```

Specified By:

[getMin\(\)](#) in interface [IIntegerMinMax](#)

getType()

```
public int getType()
```

Specified By:

[getType\(\)](#) in interface [IRuntimeNode](#)

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

hasMax()

```
public boolean hasMax()
```

Specified By:

[hasMax\(\)](#) in interface [IIntegerMinMax](#)

hasMin()

```
public boolean hasMin()
```

Specified By:

[hasMin\(\)](#) in interface [IIntegerMinMax](#)

oracle.apps.cz.cio IntegerNode

Syntax

public abstract class IntegerNode extends [RuntimeNode](#) implements [IInteger](#)

```
java.lang.Object
|
+--RuntimeNode
    |
    +--oracle.apps.cz.cio.IntegerNode
```

Direct Known Subclasses:

[IntegerFeature](#)

All Implemented Interfaces:

[IInteger](#), [IRuntimeNode](#)

Description

Represents a feature with an integer value.

Member Summary

Methods

[getIntValue\(\)](#)

[isUnknown\(\)](#)

[setIntValue\(int\)](#)

[unset\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getIntValue()

```
public int getIntValue()
```

Specified By:

[getIntValue\(\)](#) in interface [IInteger](#)

isUnknown()

```
public boolean isUnknown()
```

Specified By:

[isUnknown\(\)](#) in interface [IInteger](#)

setIntValue(int)

```
public void setIntValue(int newIntValue)
```

Specified By:

[setIntValue\(int\)](#) in interface [IInteger](#)

unset()

```
public void unset()
```

Specified By:

[unset\(\)](#) in interface [Integer](#)

oracle.apps.cz.cio

IOption

Syntax

public interface IOption extends [IState](#), [ICount](#)

All Known Subinterfaces:

[IBomItem](#)

All Superinterfaces:

[ICount](#), [IState](#)

All Known Implementing Classes:

[OptionNode](#)

Description

Implemented by objects that act as options. The defining characteristic of an option is that it can be selected and deselected.

Member Summary

Methods

deselect()	Deslect this option.
isOptionMutexed()	Returns true if this option is a child of a mutexed parent
isSelected()	Returns <code>true</code> if this option is selected, and <code>false</code> otherwise.
select()	Select this option.

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Methods inherited from interface [IState](#)

Inherited Member Summary

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods**deselect()**

```
public void deselect()  
Deselect this option.
```

isOptionMutexed()

```
public boolean isOptionMutexed()  
Returns true if this option is a child of a mutexed parent
```

isSelected()

```
public boolean isSelected()  
Returns true if this option is selected, and false otherwise.
```

select()

```
public void select()  
Select this option.
```

oracle.apps.cz.cio

IOptionFeature

Syntax

public interface IOptionFeature extends [IState](#), [ICount](#)

All Known Subinterfaces:

[IBomItem](#)

All Superinterfaces:

[ICount](#), [IState](#)

All Known Implementing Classes:

[OptionFeatureNode](#)

Description

Implemented by objects that contain selectable options. This interface provides a mechanism for selecting and deselecting options, and for determining which options are currently selected.

Member Summary

Methods

areOptionsCounted()	Returns true if the Feature has counted options.
deselect(IOption)	Deselect a particular option.
getMaxSelected()	Returns the maximum number of selected options.
getMinSelected()	Returns the minimum number of selected options.
getSelectedOption()	Returns the currently selected option, or <code>null</code> if no option is selected.
getSelectedOptions()	Returns a, possibly empty, collection of options that are currently selected.
hasMaxSelected()	Returns true if the Feature specifies a maximum number of selected options.
hasMinSelected()	Returns true if the Feature specifies a minimum number of selected options.
isSelectionMutexed()	Returns <code>true</code> if this feature supports mutexed selections.

Member Summary

select(IOption)	Select a particular option.
---------------------------------	-----------------------------

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods**areOptionsCounted()**

```
public boolean areOptionsCounted()  
Returns true if the Feature has counted options.
```

deselect(IOption)

```
public void deselect(IOption option)  
Deselect a particular option.
```

Parameters:

`option` - the option to be de selected.

getMaxSelected()

```
public int getMaxSelected()  
Returns the maximum number of selected options.
```

getMinSelected()

```
public int getMinSelected()  
Returns the minimum number of selected options.
```

getSelectedOption()

```
public IOption getSelectedOption()
```

Returns the currently selected option, or null if no option is selected.

Returns:

the currently selected option.

Throws:

[SelectionNotMutexedException](#) - if this feature does not support mutexed selections.

getSelectedOptions()

```
public java.util.List getSelectedOptions()
```

Returns a, possibly empty, collection of options that are currently selected.

hasMaxSelected()

```
public boolean hasMaxSelected()
```

Returns true if the Feature specifies a maximum number of selected options.

hasMinSelected()

```
public boolean hasMinSelected()
```

Returns true if the Feature specifies a minimum number of selected options.

isSelectionMutexed()

```
public boolean isSelectionMutexed()
```

Returns true if this feature supports mutexed selections. When a selection is mutexed, it means that only one of a particular option is selectable at any one time, and selecting one option automatically deselects any other option that is mutexed and currently selected.

select(IOption)

```
public void select(IOption option)
```

Select a particular option.

Parameters:

option - the option to be selected.

oracle.apps.cz.cio

IPrice

Syntax

public interface IPrice extends [ICount](#)

All Known Subinterfaces:

[IBomItem](#)

All Superinterfaces:

[ICount](#)

All Known Implementing Classes:

[StateCountNode](#)

Description

Implemented by objects that can be priced. This interface contains methods for getting list, discount, and extended prices, and methods to retrieve pricing errors/warnings/messages.

Member Summary

Methods

getDatabaseID()	Returns the database ID of the runtime node.
getDiscountedPrice()	Gets discounted price of item based on adjustments associated with price list specified in initParameters.
getExtendedPrice()	Calculates extended price of item (quantity * discounted price).
getItemKey()	Returns item key for items imported from Oracle Inventory / BOM.
getListPrice()	Gets list price of item on price list specified in initParameters.
getPricingNotifications()	Returns string containing any pricing messages, warnings, or errors.
getUomCode()	Returns unit of measure code for items imported from Oracle Inventory / BOM.

Inherited Member Summary

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods

getDatabaseID()

```
public int getDatabaseID()
```

Returns the database ID of the runtime node.

getDiscountedPrice()

```
public double getDiscountedPrice()
```

Gets discounted price of item based on adjustments associated with price list specified in `initParameters`.

getExtendedPrice()

```
public double getExtendedPrice()
```

Calculates extended price of item (quantity * discounted price).

getItemKey()

```
public java.lang.String getItemKey()
```

Returns item key for items imported from Oracle Inventory / BOM. Item key is constructed from BOM_EXPLOSIONS field values: "[COMPONENT_CODE]:[EXPLOSION_TYPE]:[ORGANIZATION_ID]:[TOP_ITEM_ID]" Item key may be used by PL/SQL pricing procedures to price nodes. Returns null if node was not imported from Oracle Inventory / BOM.

getListPrice()

```
public double getListPrice()
```

Gets list price of item on price list specified in `initParameters`.

getPricingNotifications()

```
public java.lang.String getPricingNotifications()
```

Returns string containing any pricing messages, warnings, or errors.

getUomCode()

```
public java.lang.String getUomCode()
```

Returns unit of measure code for items imported from Oracle Inventory / BOM. The unit of measure may be used by PL/SQL pricing procedures to price nodes. Returns null if node was not imported from Oracle Inventory / BOM.

oracle.apps.cz.cio ReadOnlyDecimal

Syntax

```
public interface ReadOnlyDecimal
```

All Known Subinterfaces:

[IDecimal](#)

All Known Implementing Classes:

[ReadOnlyDecimalNode](#)

Description

Implemented by objects that have a decimal value.

Member Summary

Methods

getDecimalValue()	Gets the current value of this object.
isUnknown()	Tells whether this feature is in unknown or known state.

Methods

getDecimalValue()

```
public double getDecimalValue()
Gets the current value of this object.
```

Returns:

the current value.

isUnknown()

```
public boolean isUnknown()-
Tells whether this feature is in unknown or known state.
```

Returns:

true if the feature is unknown.

oracle.apps.cz.cio
IRuntimeNode

Syntax

public interface IRuntimeNode

All Known Implementing Classes:

[RuntimeNode](#)

Description

Implemented by all objects in the runtime configuration tree. This interface implements behavior common to all nodes in the runtime configuration tree, including components, features, options, totals, etc.

Member Summary

Fields

ALL_FEATURES	A pseudo-type that represents all feature types for use in getChildrenByType.
BOM_MODEL	BOM model type.
BOM_OPTION_CLASS	BOM option class type.
BOM_STD_ITEM	BOM standard item type.
BOOLEAN_FEATURE	Boolean feature type.
COMPONENT	Component type.
COMPONENT_SET	Component set type.
COUNT_FEATURE	Count feature type.
DECIMAL_FEATURE	Decimal feature type.
INTEGER_FEATURE	Integer feature type.
OPTION	Option type.
OPTION_FEATURE	Option feature type.
RESOURCE	Resource type.

Member Summary

<code>TEXT_FEATURE</code>	Text feature type.
<code>TOTAL</code>	Total type.
Methods	
<code>getChildById(int)</code>	Gets a particular child identified by its ID.
<code>getChildByName(String)</code>	Gets a particular child identified by its name.
<code>getChildren()</code>	Gets the children of this runtime configuration node.
<code>getChildrenByType(int)</code>	Gets all of the children of a particular type.
<code>getConfiguration()</code>	Gets the configuration to which this node belongs.
<code>getDatabaseID()</code>	Gets the database ID of the node.
<code>getDescription()</code>	Returns the design-time description of the runtime node.
<code>getName()</code>	Gets the name of the node.
<code>getParent()</code>	Get the parent of this node.
<code>getProperties()</code>	Returns a collection of the properties associated with this node.
<code>getPropertyByName(String)</code>	Returns a particular property of this node, based on its name.
<code>getRuntimeID()</code>	Gets the runtime ID of the node.
<code>getSelectionLineID()</code>	Returns selection line ID (configuration output database ID) for node.
<code>getType()</code>	Gets the type of this node.
<code>hasCount()</code>	Returns true if the node has an object count.
<code>hasDecimalValue()</code>	Returns true if the node has a decimal value.
<code>hasDescription()</code>	Returns true if there is a design-time description of the runtime node.
<code>hasSelectionLineID()</code>	Returns true if node has a selection line ID (configuration output ID), false if not.
<code>hasState()</code>	Returns true if the node has a logical state.
<code>hasTextValue()</code>	Returns true if the node has a text value.
<code>isNative()</code>	Returns true if this is a native BOM node
<code>isUnsatisfied()</code>	Returns <code>true</code> if this particular node, or any one of its children, has not been completely configured.
<code>isUnsatisfiedNode()</code>	Returns <code>true</code> if this particular node has not been completely configured.

Member Summary

toString(boolean)	Returns a String representation of this node, based on whether the client demands a description (if there is one) or just a name
-----------------------------------	--

Fields

ALL_FEATURES

public static final int ALL_FEATURES
A pseudo-type that represents all feature types for use in getChildrenByType.

BOM_MODEL

public static final int BOM_MODEL
BOM model type.

BOM_OPTION_CLASS

public static final int BOM_OPTION_CLASS
BOM option class type.

BOM_STD_ITEM

public static final int BOM_STD_ITEM
BOM standard item type.

BOOLEAN_FEATURE

public static final int BOOLEAN_FEATURE
Boolean feature type.

COMPONENT

public static final int COMPONENT
Component type.

COMPONENT_SET

public static final int COMPONENT_SET
Component set type.

COUNT_FEATURE

public static final int COUNT_FEATURE

Count feature type.

DECIMAL_FEATURE

public static final int DECIMAL_FEATURE
Decimal feature type.

INTEGER_FEATURE

public static final int INTEGER_FEATURE
Integer feature type.

OPTION

public static final int OPTION
Option type.

OPTION_FEATURE

public static final int OPTION_FEATURE
Option feature type.

RESOURCE

public static final int RESOURCE
Resource type.

TEXT_FEATURE

public static final int TEXT_FEATURE
Text feature type.

TOTAL

public static final int TOTAL
Total type.

Methods

getChildByID(int)

public [IRuntimeNode](#) getChildByID(int id)
Gets a particular child identified by its ID.

Returns:

a child of this node.

getChildByName(String)

public [IRuntimeNode](#) getChildByName(java.lang.String name)
Gets a particular child identified by its name.

Returns:

a child of this node.

getChildren()

public java.util.List getChildren()
Gets the children of this runtime configuration node.

Returns:

a collection of children.

getChildrenByType(int)

public java.util.List getChildrenByType(int type)
Gets all of the children of a particular type.

Returns:

a collection of children.

getConfiguration()

public [Configuration](#) getConfiguration()
Gets the configuration to which this node belongs.

Returns:

the configuration to which this node belongs.

See Also:

[Configuration](#)

getDatabaseID()

public int getDatabaseID()

Gets the database ID of the node.

Returns:

the database ID of the node.

getDescription()

public java.lang.String getDescription()

Returns the design-time description of the runtime node.

getName()

public java.lang.String getName()

Gets the name of the node.

Returns:

the name of the node.

getParent()

public IRuntimeNode getParent()

Get the parent of this node.

Returns:

the node's parent.

getProperties()

public java.util.Collection getProperties()

Returns a collection of the properties associated with this node. The collection contains items of the type Property.

getPropertyByName(String)

public Property getPropertyByName(java.lang.String name)

Returns a particular property of this node, based on its name. Returns null if a property of the given name does not exist.

getRuntimeID()

public int getRuntimeID()

Gets the runtime ID of the node. This ID is unique across all other nodes created by a particular CIO.

Returns:

runtime ID of the node.

getSelectionLineID()

```
public int getSelectionLineID()
```

Returns selection line ID (configuration output database ID) for node. The hasSelectionLineID() method should always be called before this method. A RuntimeException will be thrown if ID doesn't exist.

Returns:

line ID

getType()

```
public int getType()
```

Gets the type of this node.

Returns:

the type of this node.

hasCount()

```
public boolean hasCount()
```

Returns true if the node has an object count.

hasDecimalValue()

```
public boolean hasDecimalValue()
```

Returns true if the node has a decimal value.

hasDescription()

```
public boolean hasDescription()
```

Returns true if there is a design-time description of the runtime node.

hasSelectionLineID()

```
public boolean hasSelectionLineID()
```

Returns true if node has a selection line ID (configuration output ID), false if not.

hasState()

```
public boolean hasState()
```

Returns true if the node has a logical state.

hasTextValue()

```
public boolean hasTextValue()
```

Returns true if the node has a text value.

isNative()

```
public boolean isNative()
```

Returns true if this is a native BOM node

isUnsatisfied()

```
public boolean isUnsatisfied()
```

Returns true if this particular node, or any one of its children, has not been completely configured. The value is cached and is only updated on transaction commit or rollback.

Returns:

a boolean indicating whether the node is unsatisfied.

isUnsatisfiedNode()

```
public boolean isUnsatisfiedNode()
```

Returns true if this particular node has not been completely configured. The value is cached and is only updated on transaction commit or rollback.

Returns:

a boolean indicating whether the node is unsatisfied.

toString(boolean)

```
public java.lang.String toString(boolean description)
```

Returns a String representation of this node, based on whether the client demands a description (if there is one) or just a name

oracle.apps.cz.cio

IState

Syntax

public interface IState

All Known Subinterfaces:

[IBomItem](#), [IOption](#), [IOptionFeature](#)

All Known Implementing Classes:

[StateNode](#)

Description

Implemented by objects that have logic state. This interface contains a set of input states, used to specify a new state for an object, a set of output states, returned when querying an object for its state, and a set of methods for getting and setting the object's state.

Member Summary

Fields

FALSE	The input state used to set an object to false.
LFALSE	The logically false output state, indicating that the state is false as a consequence of a rule.
LTRUE	The logically true output state, indicating that the state is true as a consequence of a rule.
TOGGLE	The input state used to turn an object state to true if it is false or unknown, and to make it unknown or false if it is true.
TRUE	The input state used to set an object to true.
UFALSE	The user false output state, indicating that a user has set this object to false.
UNKNOWN	The unknown output state.
UTRUE	The user true output state, indicating that a user has set this object to true.

Methods

Member Summary

<code>getState()</code>	Gets the current logic state of this object.
<code>isUnknown()</code>	Tells whether this feature is in unknown or known state.
<code>setState(int)</code>	Change the current logic state of this object.
<code>unset()</code>	Retracts any user selection made toward this node

Fields**FALSE**

```
public static final int FALSE
```

The input state used to set an object to false.

LFALSE

```
public static final int LFALSE
```

The logically false output state, indicating that the state is false as a consequence of a rule.

LTRUE

```
public static final int LTRUE
```

The logically true output state, indicating that the state is true as a consequence of a rule.

TOGGLE

```
public static final int TOGGLE
```

The input state used to turn an object state to true if it is false or unknown, and to make it unknown or false if it is true.

TRUE

```
public static final int TRUE
```

The input state used to set an object to true.

UFALSE

```
public static final int UFALSE
```

The user false output state, indicating that a user has set this object to false.

UNKNOWN

```
public static final int UNKNOWN
```

The unknown output state.

UTRUE

```
public static final int UTRUE
```

The user true output state, indicating that a user has set this object to true.

Methods**getState()**

```
public int getState()
```

Gets the current logic state of this object.

Returns:

the current state.

isUnknown()

```
public boolean isUnknown()
```

Tells whether this feature is in unknown or known state.

Returns:

true if the feature is unknown.

setState(int)

```
public void setState(int state)
```

Change the current logic state of this object.

unset()

```
public void unset()
```

Retracts any user selection made toward this node

oracle.apps.cz.cio IText

Syntax

```
public interface IText
```

All Known Implementing Classes:

[TextNode](#)

Description

Implemented by objects that have a textual value.

Member Summary

Methods

getTextValue()	Gets the current textual value of this object.
setTextValue(String)	Sets the current textual value of this object.
unset()	Retracts any user selection made toward this node

Methods

getTextValue()

```
public java.lang.String getTextValue()  
Gets the current textual value of this object.
```

Returns:

the current value.

setTextValue(String)

```
public void setTextValue(java.lang.String value)  
Sets the current textual value of this object.
```

unset()

```
public void unset()
```

Retracts any user selection made toward this node

oracle.apps.cz.cio LogicalException

Syntax

```
public class LogicalException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.LogicalException
```

Direct Known Subclasses:

[LogicalOverridableException](#)

All Implemented Interfaces:

[java.io.Serializable](#)

Description

Signalled if a logical failure occurs. This failure could either be a contradiction, or a more serious problem.

Member Summary

Constructors

[LogicalException\(\)](#)

[LogicalException\(LogicalException,
Configuration\)](#)

[LogicalException\(Reason,
Configuration\)](#)

[LogicalException\(String,
Configuration\)](#)

Methods

Member Summary

[getCause\(\)](#)
[getMessage\(\)](#)
[getMessageHeader\(\)](#)
[getReasons\(\)](#)
[isOverridable\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, printStackTrace, printStackTrace, printStackTrace, toString
Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

LogicalException()

```
public LogicalException()
```

LogicalException(LogicException, Configuration)

```
public LogicalException(oracle.apps.cz.logic.LogicException le, Configuration config)
```

LogicalException(Reason, Configuration)

```
public LogicalException(Reason r, Configuration config)
```

LogicalException(String, Configuration)

```
public LogicalException(java.lang.String msg, Configuration config)
```

Methods

getCause()

```
public IRuntimeNode getCause()
```

getMessage()

```
public java.lang.String getMessage()
```

Overrides:

java.lang.Throwable.getMessage() in class java.lang.Throwable

getMessageHeader()

```
public java.lang.String getMessageHeader()
```

getReasons()

```
public java.util.List getReasons()
```

isOverridable()

```
public boolean isOverridable()
```

oracle.apps.cz.cio LogicalOverridableException

Syntax

```
public class LogicalOverridableException extends LogicalException

java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--LogicalException
            |
            +--oracle.apps.cz.cio.LogicalOverridableException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signalled if a logical contradiction occurs that can be overridden.

Member Summary
Constructors
LogicalOverridableException(LogicContradictionException, Configuration)
Methods
equals(Object)
isOverridable()
override()

Inherited Member Summary

Methods inherited from interface [LogicalException](#)

[getCause\(\)](#), [getMessage\(\)](#), [getMessageHeader\(\)](#), [getReasons\(\)](#)

Methods inherited from class java.lang.Throwable

[fillInStackTrace\(\)](#), [getLocalizedMessage\(\)](#), [printStackTrace\(\)](#), [printStackTrace\(\)](#), [printStackTrace\(\)](#), [toString\(\)](#)

Methods inherited from class java.lang.Object

[getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(\)](#), [wait\(\)](#)

Constructors

LogicalOverridableException(LogicContradictionException, Configuration)

```
public  
LogicalOverridableException(oracle.apps.cz.logic.LogicContradictionException  
lce, Configuration config)
```

Methods

equals(Object)

```
public boolean equals(java.lang.Object lce)
```

Overrides:

java.lang.Object.equals(java.lang.Object) in class java.lang.Object

isOverridable()

```
public boolean isOverridable()
```

Overrides:

[isOverridable\(\)](#) in class [LogicalException](#)

override()

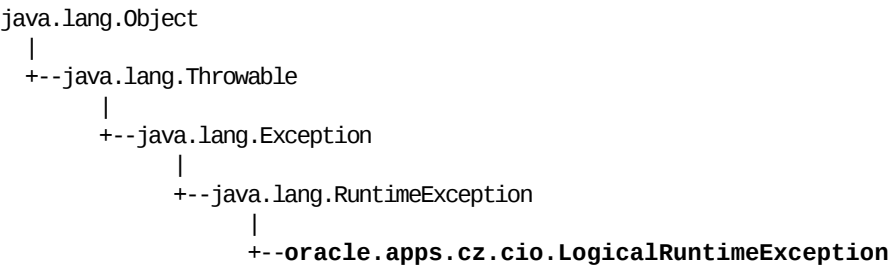
```
public void override()
```

oracle.apps.cz.cio

LogicalRuntimeException

Syntax

public class LogicalRuntimeException extends java.lang.RuntimeException



All Implemented Interfaces:

java.io.Serializable

Description

Signalled if a fatal logic exception occurred.

Member Summary

Constructors

[LogicalRuntimeException\(LogicException\)](#)

[LogicalRuntimeException\(String\)](#)

Methods

[getLogicException\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

Inherited Member Summary

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors**LogicalRuntimeException(LogicException)**

```
public LogicalRuntimeException(oracle.apps.cz.logic.LogicException le)
```

LogicalRuntimeException(String)

```
public LogicalRuntimeException(java.lang.String msg)
```

Methods**getLogicException()**

```
public oracle.apps.cz.logic.LogicException getLogicException()
```

oracle.apps.cz.cio MissingFileException

Syntax

```
public class MissingFileException extends java.lang.RuntimeException
```

```
java.lang.Object
```

```
|
```

```
+--java.lang.Throwable
```

```
|
```

```
+--java.lang.Exception
```

```
|
```

```
+--java.lang.RuntimeException
```

```
|
```

```
+--oracle.apps.cz.cio.MissingFileException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Signalled if a particular logic file is missing.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio MissingLogicException

Syntax

```
public class MissingLogicException extends java.lang.RuntimeException
```

```
java.lang.Object
```

```
|
```

```
+--java.lang.Throwable
```

```
|
```

```
+--java.lang.Exception
```

```
|
```

```
+--java.lang.RuntimeException
```

```
|
```

```
+--oracle.apps.cz.cio.MissingLogicException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Signalled if a particular logic record is missing.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio NoAtpCalculatedException

Syntax

```
public class NoAtpCalculatedException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--oracle.apps.cz.cio.NoAtpCalculatedException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Exception which is thrown when an ATP method is called on an item for which ATP is not calculated.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio NoConfigHeaderException

Syntax

```
public class NoConfigHeaderException extends java.lang.Exception

java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.NoConfigHeaderException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signalled if the configuration hasn't been saved yet.

Member Summary

Constructors

[NoConfigHeaderException\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

NoConfigHeaderException()

```
public NoConfigHeaderException()
```

oracle.apps.cz.cio NonPricedException

Syntax

```
public class NonPricedException extends java.lang.Exception
```

```
java.lang.Object
```

```
|
```

```
+--java.lang.Throwable
```

```
|
```

```
+--java.lang.Exception
```

```
|
```

```
+--oracle.apps.cz.cio.NonPricedException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Exception which is thrown when a pricing method is called on an item which should not be priced.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio NoSuchChildException

Syntax

```
public class NoSuchChildException extends java.lang.Exception
```

```
java.lang.Object
|
+--java.lang.Throwable
    |
    +--java.lang.Exception
        |
        +--oracle.apps.cz.cio.NoSuchChildException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Signalled if a requested child does not exist.

Member Summary

Constructors

[NoSuchChildException\(IRuntimeNode, int\)](#)

[NoSuchChildException\(IRuntimeNode, String\)](#)

Methods

[getID\(\)](#)

[getName\(\)](#)

[getParent\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

NoSuchChildException(IRuntimeNode, int)

```
public NoSuchChildException(IRuntimeNode parent, int id)
```

NoSuchChildException(IRuntimeNode, String)

```
public NoSuchChildException(IRuntimeNode parent, java.lang.String name)
```

Methods

getID()

```
public int getID()
```

getName()

```
public java.lang.String getName()
```

getParent()

```
public IRuntimeNode getParent()
```

oracle.apps.cz.cio NotOneProductException

Syntax

```
public class NotOneProductException extends java.lang.Exception
```

```
java.lang.Object
```

```
|
```

```
+--java.lang.Throwable
```

```
|
```

```
+--java.lang.Exception
```

```
|
```

```
+--oracle.apps.cz.cio.NotOneProductException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Exception which is thrown when a client tries to create a configuration by specifying the name of the project and the project contains more than one or no products.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio NotOneProjectException

Syntax

```
public class NotOneProjectException extends java.lang.Exception
```

```
java.lang.Object
```

```
|
```

```
+--java.lang.Throwable
```

```
|
```

```
+--java.lang.Exception
```

```
|
```

```
+---oracle.apps.cz.cio.NotOneProjectException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Exception which is thrown when a client tries to create a configuration by specifying the name of the project and the project name identifies more than one or no projects.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

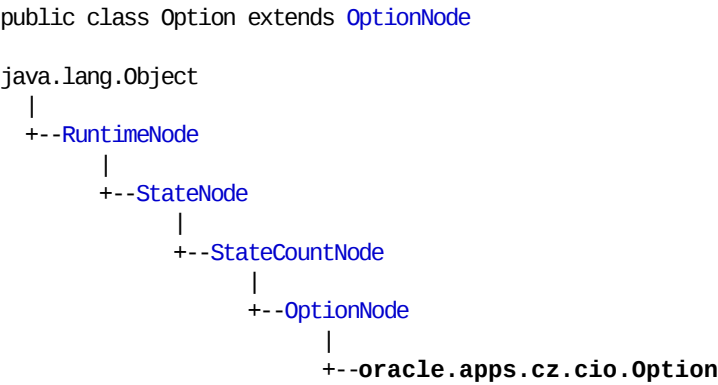
Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

oracle.apps.cz.cio

Option

Syntax



All Implemented Interfaces:

[ICount](#), [IOption](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents an option of an option feature.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [OptionNode](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#), [setState\(int\)](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IOption](#)

[deselect\(\)](#), [isOptionMutexed\(\)](#), [isSelected\(\)](#), [select\(\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio OptionFeature

Syntax

public class OptionFeature extends [OptionFeatureNode](#)

java.lang.Object

|
+--[RuntimeNode](#)

|
+--[StateNode](#)

|
+--[StateCountNode](#)

|
+--[OptionFeatureNode](#)

|
+--**oracle.apps.cz.cio.OptionFeature**

All Implemented Interfaces:

[ICount](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Represents a feature with selectable options.

Member Summary

Methods

[areOptionsCounted\(\)](#)

[getMaxSelected\(\)](#)

[getMinSelected\(\)](#)

[getType\(\)](#)

[hasMaxSelected\(\)](#)

[hasMinSelected\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [OptionFeatureNode](#)

[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [setState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IOptionFeature](#)

[deselect\(IOption\)](#), [getSelectedOption\(\)](#), [getSelectedOptions\(\)](#), [isSelectionMutexed\(\)](#), [select\(IOption\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IPrice](#)

Inherited Member Summary

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods**areOptionsCounted()**

public boolean areOptionsCounted()

getMaxSelected()

public int getMaxSelected()

getMinSelected()

public int getMinSelected()

getType()

public int getType()

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

hasMaxSelected()

public boolean hasMaxSelected()

hasMinSelected()

public boolean hasMinSelected()

oracle.apps.cz.cio

OptionFeatureNode

Syntax

public abstract class OptionFeatureNode extends [StateCountNode](#) implements [IOptionFeature](#)



Direct Known Subclasses:

[BomNode](#), [OptionFeature](#)

All Implemented Interfaces:

[ICount](#), [IOptionFeature](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

An abstract class implementing behavior commont to all features with options.

Member Summary

Methods

- [deselect\(IOption\)](#)
 - [getSelectedOption\(\)](#)
 - [getSelectedOptions\(\)](#)
 - [isSelectionMutexed\(\)](#)
 - [select\(IOption\)](#)
-

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [setState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IOptionFeature](#)

[areOptionsCounted\(\)](#), [getMaxSelected\(\)](#), [getMinSelected\(\)](#), [hasMaxSelected\(\)](#), [hasMinSelected\(\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

deselect(IOption)

```
public void deselect(IOption option)
```

Specified By:

[deselect\(IOption\)](#) in interface [IOptionFeature](#)

getSelectedOption()

```
public IOption getSelectedOption()
```

Specified By:

[getSelectedOption\(\)](#) in interface [IOptionFeature](#)

getSelectedOptions()

```
public java.util.List getSelectedOptions()
```

Specified By:

[getSelectedOptions\(\)](#) in interface [IOptionFeature](#)

isSelectionMutexed()

```
public boolean isSelectionMutexed()
```

Specified By:

[isSelectionMutexed\(\)](#) in interface [IOptionFeature](#)

select(IOption)

```
public void select(IOption option)
```

Specified By:

`select(IOption)` in interface `IOptionFeature`

oracle.apps.cz.cio OptionNode

Syntax

public abstract class OptionNode extends [StateCountNode](#) implements [IOption](#)

java.lang.Object

|

+--[RuntimeNode](#)

|

+--[StateNode](#)

|

+--[StateCountNode](#)

|

+--[oracle.apps.cz.cio.OptionNode](#)

Direct Known Subclasses:

[Option](#)

All Implemented Interfaces:

[ICount](#), [IOption](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

An abstract class implementing behavior common to all option-like objects.

Member Summary

Methods

[deselect\(\)](#)

[isOptionMutexed\(\)](#)

[isSelected\(\)](#)

[select\(\)](#)

[setState\(int\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [StateCountNode](#)

[addPricingNotification\(String\)](#), [clearDiscountedPrice\(\)](#), [clearPricingNotifications\(\)](#), [getCount\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#), [setCount\(int\)](#), [setDiscountedPrice\(double\)](#), [setListPrice\(double\)](#), [toString\(\)](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [unset\(\)](#)

Methods inherited from interface [ICount](#)

[getCount\(\)](#), [setCount\(int\)](#), [unset\(\)](#)

Methods inherited from interface [IPrice](#)

[getDatabaseID\(\)](#), [getDiscountedPrice\(\)](#), [getExtendedPrice\(\)](#), [getItemKey\(\)](#), [getListPrice\(\)](#), [getPricingNotifications\(\)](#), [getUomCode\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

deselect()

```
public void deselect()
```

Specified By:

[deselect\(\)](#) in interface [IOption](#)

isOptionMutexed()

```
public boolean isOptionMutexed()
```

Specified By:

[isOptionMutexed\(\)](#) in interface [IOption](#)

isSelected()

```
public boolean isSelected()
```

Specified By:

[isSelected\(\)](#) in interface [IOption](#)

select()

```
public void select()
```

Specified By:

[select\(\)](#) in interface [IOption](#)

setState(int)

```
public void setState(int newState)
```

Specified By:

[setState\(int\)](#) in interface [IState](#)

Specified By:

[setState\(int\)](#) in interface [IState](#)

Overrides:

[setState\(int\)](#) in class [StateNode](#)

oracle.apps.cz.cio

PricingUnavailableException

Syntax

```
public class PricingUnavailableException extends java.lang.Exception

java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.PricingUnavailableException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signals that the CIO pricing functionality is not available.

Member Summary

Constructors

- [PricingUnavailableException\(String\)](#)
 - [PricingUnavailableException\(String, Object, Log\)](#)
-

Inherited Member Summary

- Methods inherited from class java.lang.Throwable
- fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString
- Methods inherited from class java.lang.Object
- equals, getClass, hashCode, notify, notifyAll, wait, wait, wait
-

Constructors

PricingUnavailableException(String)

```
public PricingUnavailableException(java.lang.String reason)
```

PricingUnavailableException(String, Object, Log)

```
public PricingUnavailableException(java.lang.String reason, java.lang.Object  
source, oracle.apps.fnd.common.Log log)
```

oracle.apps.cz.cio

Property

Syntax

```
public class Property extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.cio.Property
```

Description

Represents name/value properties associated with runtime nodes.

Member Summary

Methods

getBooleanValue()	Returns the property's value as a boolean.
getDecimalValue()	Returns the property's value as a double.
getDescription()	Returns the property's description.
getIntValue()	Returns the property's value as an integer.
getName()	Returns the property's name.
getStringValue()	Returns the property's value as a string.
getUnit()	Returns the property's unit of measure.
hasBooleanValue()	Returns true if property is a boolean property.
hasDecimalValue()	Returns true if property is a decimal property.
hasDefaultValue()	Checks to see if property has overridden its default value.
hasIntegerValue()	Returns true if property is an integer property.
hasStringValue()	Returns true if property is a string property.

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods**getBooleanValue()**

```
public boolean getBooleanValue()  
Returns the property's value as a boolean.
```

getDecimalValue()

```
public double getDecimalValue()  
Returns the property's value as a double.
```

getDescription()

```
public java.lang.String getDescription()  
Returns the property's description.
```

getIntValue()

```
public int getIntValue()  
Returns the property's value as an integer.
```

getName()

```
public java.lang.String getName()  
Returns the property's name.
```

getStringValue()

```
public java.lang.String getStringValue()  
Returns the property's value as a string.
```

getUnit()

```
public java.lang.String getUnit()  
Returns the property's unit of measure.
```

hasBooleanValue()

```
public boolean hasBooleanValue()
```

Returns true if property is a boolean property.

hasDecimalValue()

```
public boolean hasDecimalValue()
```

Returns true if property is a decimal property.

hasDefaultValue()

```
public boolean hasDefaultValue()
```

Checks to see if property has overridden its default value.

hasIntegerValue()

```
public boolean hasIntegerValue()
```

Returns true if property is an integer property.

hasStringValue()

```
public boolean hasStringValue()
```

Returns true if property is a string property.

oracle.apps.cz.cio ReadOnlyDecimalNode

Syntax

public abstract class ReadOnlyDecimalNode extends [RuntimeNode](#) implements [IRReadOnlyDecimal](#)



Direct Known Subclasses:

[DecimalNode](#)

All Implemented Interfaces:

[IRReadOnlyDecimal](#), [IRuntimeNode](#)

Description

An abstract class implementing behavior common to objects with a decimal value.

Member Summary

Methods

[getDecimalValue\(\)](#)

[isUnknown\(\)](#)

[toString\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getDecimalValue()

```
public double getDecimalValue()
```

Specified By:

[getDecimalValue\(\)](#) in interface [IReadingOnlyDecimal](#)

isUnknown()

```
public boolean isUnknown()
```

Specified By:

[isUnknown\(\)](#) in interface [IReadingOnlyDecimal](#)

toString()

```
public java.lang.String toString()
```

Overrides:

[toString\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio Reason

Syntax

```
public class Reason extends java.lang.Object
```

```
java.lang.Object
|
+--oracle.apps.cz.cio.Reason
```

Description

This class wraps the information returned by a contradiction in order to include information about internal error messages.

Member Summary

Fields

DEFAULT	This reason initiated from inability to set a state, because of a default relation.
INTL_TEXT	The message is an internationalized text string.
MINMAX	This reason initiated from an internal MINMAX relationship.
ORTHEN	This reason initiated from an internal ORTHEN relationship.
TEXT	The message is an unknown format text string.
TRUEATBIRTH	This reason initiated from an internal relationship for a group.

Constructors

Reason(int, IRuntimeNode, String)	Construct a reason given all of it's information.
Reason(Message, String, IRuntimeNode)	Constructs a reason from an FND message.
Reason(String)	Constructs a simple TEXT reason.

Methods

getFndMessageString(Message)	
getMsg()	Get the message associated with this reason.

Member Summary

getNode()	Get the node associated with this reason.
getType()	Get the type of reason is held in this object.
toString()	Convert this object to a string.
translate()	This method returns the translated string for the reason.
translate(String)	This method returns the translated reason string using the given name for substitution variable.

Inherited Member Summary

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Fields

DEFAULT

`public static int DEFAULT`
This reason initiated from inability to set a state, because of a default relation.

INTL_TEXT

`public static int INTL_TEXT`
The message is an internationalized text string.

MINMAX

`public static int MINMAX`
This reason initiated from an internal MINMAX relationship.

ORTHEN

`public static int ORTHEN`
This reason initiated from an internal ORTHEN relationship.

TEXT

`public static int TEXT`

The message is an unknown format text string.

TRUEATBIRTH

```
public static int TRUEATBIRTH
```

This reason initiated from an internal relationship for a group.

Constructors

Reason(int, IRuntimeNode, String)

```
public Reason(int type, IRuntimeNode node, java.lang.String msg)
```

Construct a reason given all of it's information.

Parameters:

type - What type of reason this is.

node - The node that caused the problem.

msg - The message returned.

Reason(Message, String, IRuntimeNode)

```
public Reason(oracle.apps.fnd.common.Message fndMsg, java.lang.String token,  
IRuntimeNode node)
```

Constructs a reason from an FND message.

Parameters:

fndMsg - The FND message object with all but one token substituted.

token - The token name left to substitute.

node - The node requiring substitution.

Reason(String)

```
public Reason(java.lang.String msg)
```

Constructs a simple TEXT reason.

Parameters:

msg - The message string for the reason.

Methods

getFndMessageString(Message)

```
public java.lang.String getFndMessageString(oracle.apps.fnd.common.Message  
fndMsg)
```

getMsg()

```
public java.lang.String getMsg()  
Get the message associated with this reason.
```

getNode()

```
public IRuntimeNode getNode()  
Get the node associated with this reason.
```

getType()

```
public int getType()  
Get the type of reason is held in this object.
```

toString()

```
public java.lang.String toString()  
Convert this object to a string.
```

Overrides:

`java.lang.Object.toString()` in class `java.lang.Object`

translate()

```
public java.lang.String translate()  
This method returns the translated string for the reason. If the string has a node  
name substitution then the internal name is used.
```

translate(String)

```
public java.lang.String translate(java.lang.String nodeName)  
This method returns the translated reason string using the given name for  
substitution variable.
```

Parameters:

`nodeName` - The node name to substitute into the string.

oracle.apps.cz.cio Resource

Syntax

public class Resource extends [DecimalNode](#)

```
java.lang.Object
|
+--RuntimeNode
    |
    +--ReadOnlyDecimalNode
        |
        +--DecimalNode
            |
            +--oracle.apps.cz.cio.Resource
```

All Implemented Interfaces:

[IDecimal](#), [IReadOnlyDecimal](#), [IRuntimeNode](#)

Description

Represents a consumable resource. A resource will signal a validation failure when it is overconsumed (in other words, when its value goes below zero). NOTE: This class inherits from [DecimalNode](#), but the functionality of a [DecimalNode](#) (specifically the method [SetDecimalValue\(\)](#)) is 'deprecated', meaning that it shouldn't be used on new projects and may be unsupported in a future release. Use only methods inherited from [ReadOnlyDecimalNode](#).

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [DecimalNode](#)

[setDecimalValue\(double\)](#), [toString\(\)](#), [unset\(\)](#)

Methods inherited from class [ReadOnlyDecimalNode](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class java.lang.Object

[equals\(\)](#), [getClass\(\)](#), [hashCode\(\)](#), [notify\(\)](#), [notifyAll\(\)](#), [wait\(\)](#), [wait\(long\)](#), [wait\(long, int\)](#)

Methods inherited from interface [IDecimal](#)

[setDecimalValue\(double\)](#), [unset\(\)](#)

Methods inherited from interface [IRReadOnlyDecimal](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio RestoreValidationFailure

Syntax

public class RestoreValidationFailure extends [ValidationFailure](#)

java.lang.Object

|

+--[StatusInfo](#)

|

+--[ValidationFailure](#)

|

+--**oracle.apps.cz.cio.RestoreValidationFailure**

Description

Failure produced when restoring a configuration over a changed model.

Member Summary

Methods

[equals\(Object\)](#)

[getInput\(\)](#) Returns the input object where the validation failure occurred

[hashCode\(\)](#)

Inherited Member Summary

Fields inherited from class [ValidationFailure](#)

[COMPANION_FAILURE](#), [MAX_FAILURE](#), [MIN_FAILURE](#), [MIN0_FAILURE](#), [MINMAX_FAILURE](#), [RESOURCE_FAILURE](#), [RESTORE_FAILURE](#)

Fields inherited from class [StatusInfo](#)

[STATUS_DELETED](#), [STATUS_EXISTING](#), [STATUS_NEW](#)

Methods inherited from class [ValidationFailure](#)

Inherited Member Summary

[getMessage\(\)](#), [getMessage\(String\)](#), [getType\(\)](#), [toString\(\)](#)

Methods inherited from class [StatusInfo](#)

[getNode\(\)](#), [getStatus\(\)](#), [statusToString\(int\)](#), [toString\(boolean\)](#)

Methods inherited from class java.lang.Object

[getClass](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Overrides:

[equals\(Object\)](#) in class [ValidationFailure](#)

getInput()

```
public oracle.apps.cz.dio.config.DbConfigInput getInput()
```

Returns the input object where the validation failure occurred

Returns:

the failed DbConfigInput object

hashCode()

```
public int hashCode()
```

Overrides:

[hashCode\(\)](#) in class [StatusInfo](#)

oracle.apps.cz.cio RuntimeNode

Syntax

public abstract class RuntimeNode extends java.lang.Object implements [IRuntimeNode](#)

```
java.lang.Object
|
+--oracle.apps.cz.cio.RuntimeNode
```

Direct Known Subclasses:

[ComponentNode](#), [IntegerNode](#), [ReadOnlyDecimalNode](#), [StateNode](#), [TextNode](#)

All Implemented Interfaces:

[IRuntimeNode](#)

Description

Abstract class implementing common behavior across all runtime nodes.

Member Summary

Methods

getChildByID(int)	Returns the child of this node with a given database ID.
getChildByName(String)	Returns the child of this node with a given name.
getChildren()	Returns a list of all children of this runtime node.
getChildrenByType(int)	Returns a list of all children of a particular type.
getConfiguration()	Returns the configuration to which this node belongs.
getDatabaseID()	Returns the database ID of the runtime node.
getDescription()	Returns the design-time description of the runtime node.
getName()	Returns the name of the runtime node.
getParent()	Returns the parent of this runtime node, or <code>null</code> if this is the root node.

Member Summary

getProperties()	Returns a collection of the properties associated with this node.
getPropertyByName(String)	Returns a particular property of this node, based on its name.
getRuntimeID()	Returns the runtime ID for the node.
getSelectionLineID()	
getType()	Returns the type of the runtime node.
hasCount()	Returns true if the node has an object count.
hasDecimalValue()	Returns true if the node has a decimal value.
hasDescription()	Returns true if there is a design-time description of the runtime node.
hasIntegerValue()	Returns true if the node has a integer value.
hasSelectionLineID()	
hasState()	Returns true if the node has a logical state.
hasTextValue()	Returns true if the node has a text value.
isNative()	Returns true if this is a native BOM node
isUnsatisfied()	Returns true if this runtime node, or any of its children, is not fully configured.
isUnsatisfiedNode()	Returns true if this particular node is not fully configured.
lookupNodeID(Object)	
toString()	
toString(boolean)	
typeToString(int)	Returns a string representation of a given runtime node type constant.

Inherited Member Summary

Fields inherited from interface IRuntimeNode	
ALL_FEATURES , BOM_MODEL , BOM_OPTION_CLASS , BOM_STD_ITEM , BOOLEAN_FEATURE , COMPONENT , COMPONENT_SET , COUNT_FEATURE , DECIMAL_FEATURE , INTEGER_FEATURE , OPTION , OPTION_FEATURE , RESOURCE , TEXT_FEATURE , TOTAL	
Methods inherited from class java.lang.Object	
equals , getClass , hashCode , notify , notifyAll , wait , wait , wait	

Methods

getChildByID(int)

public [IRuntimeNode](#) getChildByID(int id)

Returns the child of this node with a given database ID.

Specified By:

[getChildByID\(int\)](#) in interface [IRuntimeNode](#)

Throws:

[NoSuchChildException](#) - if there is no child with such ID.

getChildByName(String)

public [IRuntimeNode](#) getChildByName(java.lang.String name)

Returns the child of this node with a given name.

Specified By:

[getChildByName\(String\)](#) in interface [IRuntimeNode](#)

Throws:

[NoSuchChildException](#) - if there is no child with such name.

getChildren()

public java.util.List getChildren()

Returns a list of all children of this runtime node.

Specified By:

[getChildren\(\)](#) in interface [IRuntimeNode](#)

getChildrenByType(int)

public java.util.List getChildrenByType(int type)

Returns a list of all children of a particular type.

Specified By:

[getChildrenByType\(int\)](#) in interface [IRuntimeNode](#)

getConfiguration()

```
public Configuration getConfiguration()  
Returns the configuration to which this node belongs.
```

Specified By:

[getConfiguration\(\)](#) in interface [IRuntimeNode](#)

getDatabaseID()

```
public int getDatabaseID()  
Returns the database ID of the runtime node.
```

Specified By:

[getDatabaseID\(\)](#) in interface [IRuntimeNode](#)

getDescription()

```
public java.lang.String getDescription()  
Returns the design-time description of the runtime node.
```

Specified By:

[getDescription\(\)](#) in interface [IRuntimeNode](#)

getName()

```
public java.lang.String getName()  
Returns the name of the runtime node.
```

Specified By:

[getName\(\)](#) in interface [IRuntimeNode](#)

getParent()

```
public IRuntimeNode getParent()  
Returns the parent of this runtime node, or null if this is the root node.
```

Specified By:

[getParent\(\)](#) in interface [IRuntimeNode](#)

getProperties()

```
public java.util.Collection getProperties()
```

Returns a collection of the properties associated with this node. The collection contains items of the type `IProperty`.

Specified By:

[getProperties\(\)](#) in interface [IRuntimeNode](#)

getPropertyByName(String)

```
public Property getPropertyByName(java.lang.String name)
```

Returns a particular property of this node, based on its name. Returns null if a property of the given name does not exist.

Specified By:

[getPropertyByName\(String\)](#) in interface [IRuntimeNode](#)

getRuntimeID()

```
public int getRuntimeID()
```

Returns the runtime ID for the node. This ID is unique across all nodes in a particular configuration.

Specified By:

[getRuntimeID\(\)](#) in interface [IRuntimeNode](#)

getSelectionLineID()

```
public int getSelectionLineID()
```

Specified By:

[getSelectionLineID\(\)](#) in interface [IRuntimeNode](#)

getType()

```
public abstract int getType()
```

Returns the type of the runtime node. Must be implemented.

Specified By:

[getType\(\)](#) in interface [IRuntimeNode](#)

hasCount()

public boolean hasCount()
Returns true if the node has an object count.

Specified By:

[hasCount\(\)](#) in interface [IRuntimeNode](#)

hasDecimalValue()

public boolean hasDecimalValue()
Returns true if the node has a decimal value.

Specified By:

[hasDecimalValue\(\)](#) in interface [IRuntimeNode](#)

hasDescription()

public boolean hasDescription()
Returns true if there is a design-time description of the runtime node.

Specified By:

[hasDescription\(\)](#) in interface [IRuntimeNode](#)

hasIntegerValue()

public boolean hasIntegerValue()
Returns true if the node has a integer value.

hasSelectionLineID()

public boolean hasSelectionLineID()

Specified By:

[hasSelectionLineID\(\)](#) in interface [IRuntimeNode](#)

hasState()

public boolean hasState()
Returns true if the node has a logical state.

Specified By:

[hasState\(\)](#) in interface [IRuntimeNode](#)

hasTextValue()

public boolean hasTextValue()
Returns true if the node has a text value.

Specified By:

[hasTextValue\(\)](#) in interface [IRuntimeNode](#)

isNative()

public boolean isNative()
Returns true if this is a native BOM node

Specified By:

[isNative\(\)](#) in interface [IRuntimeNode](#)

isUnsatisfied()

public boolean isUnsatisfied()
Returns true if this runtime node, or any of its children, is not fully configured.

Specified By:

[isUnsatisfied\(\)](#) in interface [IRuntimeNode](#)

isUnsatisfiedNode()

public boolean isUnsatisfiedNode()
Returns true if this particular node is not fully configured.

Specified By:

[isUnsatisfiedNode\(\)](#) in interface [IRuntimeNode](#)

lookupNodeID(Object)

public [IRuntimeNode](#) lookupNodeID(java.lang.Object id)

toString()

public java.lang.String toString()

Overrides:

java.lang.Object.toString() in class java.lang.Object

toString(boolean)

public java.lang.String toString(boolean description)

Specified By:

[toString\(boolean\)](#) in interface [IRuntimeNode](#)

typeToString(int)

public static java.lang.String typeToString(int type)

Returns a string representation of a given runtime node type constant.

oracle.apps.cz.cio SelectionNotMutexedException

Syntax

public class SelectionNotMutexedException extends java.lang.Exception

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.SelectionNotMutexedException
```

All Implemented Interfaces:

java.io.Serializable

Description

Signalled when an mutexed selection operation is performed on an option feature that does not support mutexed selection.

Member Summary

Methods

[getFeature\(\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods

getFeature()

```
public IOptionFeature getFeature()
```

oracle.apps.cz.cio StateCountNode

Syntax

public abstract class StateCountNode extends [StateNode](#) implements [ICount](#), [IPrice](#)

```
java.lang.Object
|
+--RuntimeNode
    |
    +--StateNode
        |
        +--oracle.apps.cz.cio.StateCountNode
```

Direct Known Subclasses:

[CountFeature](#), [OptionFeatureNode](#), [OptionNode](#)

All Implemented Interfaces:

[ICount](#), [IPrice](#), [IRuntimeNode](#), [IState](#)

Description

Abstract class implementing common behavior for nodes with a logic state and count.

Member Summary

Methods

[addPricingNotification\(String\)](#)

[clearDiscountedPrice\(\)](#)

[clearPricingNotifications\(\)](#)

[getCount\(\)](#)

[getDiscountedPrice\(\)](#)

[getExtendedPrice\(\)](#)

[getItemKey\(\)](#)

Member Summary

[getListPrice\(\)](#)

[getPricingNotifications\(\)](#)

[getUomCode\(\)](#)

[setCount\(int\)](#)

[setDiscountedPrice\(double\)](#)

[setListPrice\(double\)](#)

[toString\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [StateNode](#)

[getState\(\)](#), [isDefaultState\(int\)](#), [isFalseState\(int\)](#), [isLogicState\(int\)](#), [isTrueState\(int\)](#), [isUnknown\(\)](#), [isUnknownState\(int\)](#), [isUserState\(int\)](#), [setState\(int\)](#), [statesMatch\(int, int\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [ICount](#)

[unset\(\)](#)

Methods inherited from interface [IPrice](#)

Inherited Member Summary

[getDatabaseID\(\)](#)

Methods inherited from interface [IState](#)

[getState\(\)](#), [isUnknown\(\)](#), [setState\(int\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

addPricingNotification(String)

```
public void addPricingNotification(java.lang.String message)
```

clearDiscountedPrice()

```
public void clearDiscountedPrice()
```

clearPricingNotifications()

```
public void clearPricingNotifications()
```

getCount()

```
public int getCount()
```

Specified By:

[getCount\(\)](#) in interface [ICount](#)

Specified By:

[getCount\(\)](#) in interface [ICount](#)

getDiscountedPrice()

```
public double getDiscountedPrice()
```

Specified By:

[getDiscountedPrice\(\)](#) in interface [IPrice](#)

getExtendedPrice()

public double getExtendedPrice()

Specified By:

[getExtendedPrice\(\)](#) in interface [IPrice](#)

getItemKey()

public java.lang.String getItemKey()

Specified By:

[getItemKey\(\)](#) in interface [IPrice](#)

getListPrice()

public double getListPrice()

Specified By:

[getListPrice\(\)](#) in interface [IPrice](#)

getPricingNotifications()

public java.lang.String getPricingNotifications()

Specified By:

[getPricingNotifications\(\)](#) in interface [IPrice](#)

getUomCode()

public java.lang.String getUomCode()

Specified By:

[getUomCode\(\)](#) in interface [IPrice](#)

setCount(int)

public void setCount(int newCount)

Specified By:[setCount\(int\)](#) in interface [ICount](#)**Specified By:**[setCount\(int\)](#) in interface [ICount](#)**setDiscountedPrice(double)**

```
public void setDiscountedPrice(double discountedPrice)
```

setListPrice(double)

```
public void setListPrice(double listPrice)
```

toString()

```
public java.lang.String toString()
```

Overrides:[toString\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio

StateNode

Syntax

public abstract class StateNode extends [RuntimeNode](#) implements [IState](#)



Direct Known Subclasses:

[BooleanFeature](#), [StateCountNode](#)

All Implemented Interfaces:

[IRuntimeNode](#), [IState](#)

Description

Abstract class implementing common behavior across nodes with logic state.

Member Summary

Methods

getState()	
isDefaultState(int)	Returns true if the given state is default (not unknown, or user, or logic).
isFalseState(int)	Returns true if the given state is false (not unknown or true).
isLogicState(int)	Returns true if the given state is logic (not unknown, or user, or default).
isTrueState(int)	Returns true if the given state is true (not unknown or false).
isUnknown()	
isUnknownState(int)	Returns true if the given state is unknown (not true or false).
isUserState(int)	Returns true if the given state is user (not unknown, or logic, or default).
setState(int)	

Member Summary

[statesMatch\(int, int\)](#) Returns true if the two given states match.

[unset\(\)](#)

Inherited Member Summary

Fields inherited from interface [IState](#)

[FALSE](#), [LFALSE](#), [LTRUE](#), [TOGGLE](#), [TRUE](#), [UFALSE](#), [UNKNOWN](#), [UTRUE](#)

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getState()

```
public int getState()
```

Specified By:

[getState\(\)](#) in interface [IState](#)

isDefaultState(int)

```
public static boolean isDefaultState(int state)
```

Returns true if the given state is default (not unknown, or user, or logic).

isFalseState(int)

```
public static boolean isFalseState(int state)
```

Returns true if the given state is false (not unknown or true).

isLogicState(int)

```
public static boolean isLogicState(int state)
```

Returns true if the given state is logic (not unknown, or user, or default).

isTrueState(int)

```
public static boolean isTrueState(int state)
```

Returns true if the given state is true (not unknown or false).

isUnknown()

```
public boolean isUnknown()
```

Specified By:

[isUnknown\(\)](#) in interface [IState](#)

isUnknownState(int)

```
public static boolean isUnknownState(int state)
```

Returns true if the given state is unknown (not true or false).

isUserState(int)

```
public static boolean isUserState(int state)
```

Returns true if the given state is user (not unknown, or logic, or default).

setState(int)

```
public void setState(int newState)
```

Specified By:

[setState\(int\)](#) in interface [IState](#)

statesMatch(int, int)

public static boolean statesMatch(int inputState, int outputState)
Returns true if the two given states match.

unset()

public void unset()

Specified By:

[unset\(\)](#) in interface [IState](#)

oracle.apps.cz.cio

StatusInfo

Syntax

```
public class StatusInfo extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.cio.StatusInfo
```

Direct Known Subclasses:

[ValidationFailure](#)

Description

Contains information about a status change for a particular runtime node. The status can be STATUS_NEW, STATUS_EXISTING, or STATUS_DELETED. The condition for which this status holds depends on which list the status exists. Possibilities include validation failure, selected nodes, and unsatisfied nodes.

Member Summary

Fields

STATUS_DELETED	The node has newly lost this status since the last check.
STATUS_EXISTING	The already had this status during the last check, and it still does.
STATUS_NEW	The node has newly attained this status since the last check.

Methods

equals(Object)	
getNode()	Returns the runtime node with which this status is associated.
getStatus()	Returns the current status of the node.
hashCode()	
statusToString(int)	Return a printable representation of a status constant.
toString()	

Member Summary

[toString\(boolean\)](#)

Inherited Member Summary

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Fields**STATUS_DELETED**

public static final int STATUS_DELETED

The node has newly lost this status since the last check.

STATUS_EXISTING

public static final int STATUS_EXISTING

The already had this status during the last check, and it still does.

STATUS_NEW

public static final int STATUS_NEW

The node has newly attained this status since the last check.

Methods**equals(Object)**

public boolean equals(java.lang.Object obj)

Overrides:

java.lang.Object.equals(java.lang.Object) in class java.lang.Object

getNode()

public [IRuntimeNode](#) getNode()

Returns the runtime node with which this status is associated.

getStatus()

```
public int getStatus()
```

Returns the current status of the node.

hashCode()

```
public int hashCode()
```

Overrides:

java.lang.Object.hashCode() in class java.lang.Object

statusToString(int)

```
public static java.lang.String statusToString(int status)
```

Return a printable representation of a status constant.

toString()

```
public java.lang.String toString()
```

Overrides:

java.lang.Object.toString() in class java.lang.Object

toString(boolean)

```
public java.lang.String toString(boolean description)
```

oracle.apps.cz.cio TextFeature

Syntax

public class TextFeature extends [TextNode](#)

java.lang.Object

|

+--[RuntimeNode](#)

|

+--[TextNode](#)

|

+--**oracle.apps.cz.cio.TextFeature**

All Implemented Interfaces:

[IRuntimeNode](#), [IText](#)

Description

Represents a feature that has a textual value.

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [TextNode](#)

[getTextValue\(\)](#), [setTextValue\(String\)](#), [unset\(\)](#)

Methods inherited from class [RuntimeNode](#)

Inherited Member Summary

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IText](#)

[getTextValue\(\)](#), [setTextValue\(String\)](#), [unset\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio TextNode

Syntax

public abstract class TextNode extends [RuntimeNode](#) implements [IText](#)

```
java.lang.Object
|
+--RuntimeNode
    |
    +--oracle.apps.cz.cio.TextNode
```

Direct Known Subclasses:

[TextFeature](#)

All Implemented Interfaces:

[IRuntimeNode](#), [IText](#)

Description

Represents a feature that has a textual value.

Member Summary

Methods

[getTextValue\(\)](#)

[setTextValue\(String\)](#)

[unset\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Inherited Member Summary

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class [java.lang.Object](#)

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [getType\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getTextValue()

```
public java.lang.String getTextValue()
```

Specified By:

[getTextValue\(\)](#) in interface [IText](#)

setTextValue(String)

```
public void setTextValue(java.lang.String newTextValue)
```

Specified By:

[setTextValue\(String\)](#) in interface [IText](#)

unset()

```
public void unset()
```

Specified By:

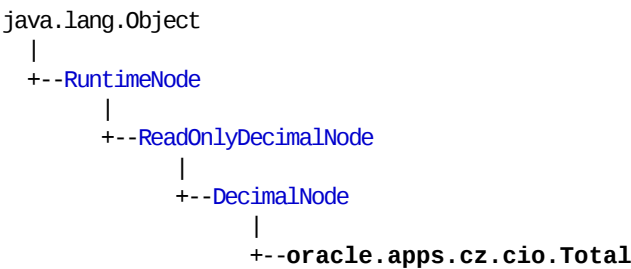
[unset\(\)](#) in interface [IText](#)

oracle.apps.cz.cio

Total

Syntax

public class Total extends [DecimalNode](#)



All Implemented Interfaces:

[IDecimal](#), [IReadOnlyDecimal](#), [IRuntimeNode](#)

Description

Represents a total that has a decimal numeric value. NOTE: This class inherits from [DecimalNode](#), but the functionality of a [DecimalNode](#) (specifically the method [SetDecimalValue\(\)](#)) is 'deprecated', meaning that it shouldn't be used on new projects and may be unsupported in a future release. Use only methods inherited from [ReadOnlyDecimalNode](#).

Member Summary

Methods

[getType\(\)](#)

Inherited Member Summary

Fields inherited from interface [IRuntimeNode](#)

Inherited Member Summary

[ALL_FEATURES](#), [BOM_MODEL](#), [BOM_OPTION_CLASS](#), [BOM_STD_ITEM](#), [BOOLEAN_FEATURE](#), [COMPONENT](#), [COMPONENT_SET](#), [COUNT_FEATURE](#), [DECIMAL_FEATURE](#), [INTEGER_FEATURE](#), [OPTION](#), [OPTION_FEATURE](#), [RESOURCE](#), [TEXT_FEATURE](#), [TOTAL](#)

Methods inherited from class [DecimalNode](#)

[setDecimalValue\(double\)](#), [toString\(\)](#), [unset\(\)](#)

Methods inherited from class [ReadOnlyDecimalNode](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from class [RuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasIntegerValue\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [lookupNodeID\(Object\)](#), [toString\(boolean\)](#), [typeToString\(int\)](#)

Methods inherited from class java.lang.Object

[equals](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Methods inherited from interface [IDecimal](#)

[setDecimalValue\(double\)](#), [unset\(\)](#)

Methods inherited from interface [IRReadOnlyDecimal](#)

[getDecimalValue\(\)](#), [isUnknown\(\)](#)

Methods inherited from interface [IRuntimeNode](#)

[getChildByID\(int\)](#), [getChildByName\(String\)](#), [getChildren\(\)](#), [getChildrenByType\(int\)](#), [getConfiguration\(\)](#), [getDatabaseID\(\)](#), [getDescription\(\)](#), [getName\(\)](#), [getParent\(\)](#), [getProperties\(\)](#), [getPropertyByName\(String\)](#), [getRuntimeID\(\)](#), [getSelectionLineID\(\)](#), [hasCount\(\)](#), [hasDecimalValue\(\)](#), [hasDescription\(\)](#), [hasSelectionLineID\(\)](#), [hasState\(\)](#), [hasTextValue\(\)](#), [isNative\(\)](#), [isUnsatisfied\(\)](#), [isUnsatisfiedNode\(\)](#), [toString\(boolean\)](#)

Methods

getType()

```
public int getType()
```

Overrides:

[getType\(\)](#) in class [RuntimeNode](#)

oracle.apps.cz.cio TransactionException

Syntax

```
public class TransactionException extends java.lang.Exception
```

```
java.lang.Object
|
+-- java.lang.Throwable
    |
    +-- java.lang.Exception
        |
        +-- oracle.apps.cz.cio.TransactionException
```

All Implemented Interfaces:

```
java.io.Serializable
```

Description

Signalled if a particular logic file is missing.

Member Summary

Methods

getAction()	Returns a String representation of the action that caused the exception
-----------------------------	---

Inherited Member Summary

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods

getAction()

```
public java.lang.String getAction()
```

Returns a String representation of the action that caused the exception

oracle.apps.cz.cio ValidationFailure

Syntax

public class ValidationFailure extends [StatusInfo](#)

java.lang.Object

|

+--[StatusInfo](#)

|

+--**oracle.apps.cz.cio.ValidationFailure**

Direct Known Subclasses:

[CompanionValidationFailure](#), [RestoreValidationFailure](#)

Description

Implements behavior common to all validation failures.

Member Summary

Fields

[COMPANION_FAILURE](#)

[MAX_FAILURE](#)

[MIN_FAILURE](#)

[MINO_FAILURE](#)

[MINMAX_FAILURE](#)

[RESOURCE_FAILURE](#)

[RESTORE_FAILURE](#)

Methods

[equals\(Object\)](#)

[getMessage\(\)](#)

[getMessage\(String\)](#)

Member Summary

[getType\(\)](#)

[toString\(\)](#)

Inherited Member Summary

Fields inherited from class [StatusInfo](#)

[STATUS_DELETED](#), [STATUS_EXISTING](#), [STATUS_NEW](#)

Methods inherited from class [StatusInfo](#)

[getNode\(\)](#), [getStatus\(\)](#), [hashCode\(\)](#), [statusToString\(int\)](#), [toString\(boolean\)](#)

Methods inherited from class java.lang.Object

[getClass](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

Fields

COMPANION_FAILURE

`public static final int COMPANION_FAILURE`

MAX_FAILURE

`public static final int MAX_FAILURE`

MIN_FAILURE

`public static final int MIN_FAILURE`

MIN0_FAILURE

`public static final int MIN0_FAILURE`

MINMAX_FAILURE

`public static final int MINMAX_FAILURE`

RESOURCE_FAILURE

`public static final int RESOURCE_FAILURE`

RESTORE_FAILURE

```
public static final int RESTORE_FAILURE
```

Methods

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Overrides:

[equals\(Object\)](#) in class [StatusInfo](#)

getMessage()

```
public java.lang.String getMessage()
```

getMessage(String)

```
public java.lang.String getMessage(java.lang.String nodeName)
```

getType()

```
public int getType()
```

toString()

```
public java.lang.String toString()
```

Overrides:

[toString\(\)](#) in class [StatusInfo](#)

Package oracle.apps.cz.common

Description

Class Summary

Classes

[CZContext](#)

Represents the runtime context of a configuration session.

oracle.apps.cz.common CZContext

Syntax

```
public final class CZContext

oracle.apps.cz.common.CZContext
```

Description

Represents the runtime context of a configuration session. The context owns the database connection, resources, and log object. It also maintains apps, user, and language information. CZContext is a shadow of the oracle.apps.fnd.common.AppsContext implementation and is designed to operate outside of the middle-tier Oracle Apps environment. The intent is for the CIO to run with either the AppsContext or CZContext depending on the runtime environment.

Member Summary

Constructors

CZContext(String, String)	Creates a new CZContext containing a connection established with the supplied database url.
CZContext(String, String, String, String)	Creates a new CZContext containing a connection established with the supplied database url, user name, and password.

Methods

createConnection(String, String, String)	Creates a new JDBC connection.
getAppId(String)	Returns the application ID for product configuration (CZ).
getCurrLangCode()	Returns the current language code.
getCurrLangInfo()	This method is restricted.
getDbOwner()	Returns name of SellingPoint schema owner.
getEnvStore()	CZContext does not contain an environment store, so this method will always throw a RuntimeException.
getJDBCConnection()	

Member Summary

getJDBCConnection(Object)	Returns the JDBC connection.
getLangCode(String)	This method is restricted.
getLangInfo(String, String)	This method is restricted.
getNLSLang(String)	This method is restricted.
getSessionManager()	This method is restricted.
getUrl()	Returns the JDBC URL.
getUser()	Returns the user name.
releaseJDBCConnection()	
setCurrLang(String)	Sets the current language code.

Constructors**CZContext(String, String)**

```
public CZContext(java.lang.String url, java.lang.String dbOwner)
```

Creates a new CZContext containing a connection established with the supplied database url.

Parameters:

url - The full url of the database.

dbOwner - the schema owner of the SellingPoint tables

CZContext(String, String, String, String)

```
public CZContext(java.lang.String url, java.lang.String user, java.lang.String password, java.lang.String dbOwner)
```

Creates a new CZContext containing a connection established with the supplied database url, user name, and password.

Parameters:

url - The full url of the database.

user - The database user name.

password - The user password.

dbOwner - the schema owner of the SellingPoint tables

Methods

createConnection(String, String, String)

```
public void createConnection(java.lang.String url, java.lang.String user,  
java.lang.String password)  
Creates a new JDBC connection.
```

getAppId(String)

```
public int getAppId(java.lang.String applShtName)  
Returns the application ID for product configuration (CZ). The parameter signature  
is for compatibility with the AppsContext implementation.
```

getCurrLangCode()

```
public java.lang.String getCurrLangCode()  
Returns the current language code.
```

getCurrLangInfo()

```
public oracle.apps.fnd.common.LangInfo getCurrLangInfo()  
This method is restricted.
```

getDbOwner()

```
public java.lang.String getDbOwner()  
Returns name of SellingPoint schema owner.
```

getEnvStore()

```
public oracle.apps.fnd.common.EnvironmentStore getEnvStore()  
CZContext does not contain an environment store, so this method will always  
throw a RuntimeException. This may need to change in the future if we depend on  
an environment store, but CZContext will only be used for testing from now on.
```

getJDBCConnection()

```
public java.sql.Connection getJDBCConnection()
```

getJDBCConnection(Object)

```
public java.sql.Connection getJDBCConnection(java.lang.Object pThis)
```

Returns the JDBC connection. The parameter signature includes a reference to the caller to be compatible with the AppsContext implementation.

getLangCode(String)

```
public java.lang.String getLangCode(java.lang.String pNLSLang)
```

This method is restricted.

getLangInfo(String, String)

```
public oracle.apps.fnd.common.LangInfo getLangInfo(java.lang.String pLangCode,  
java.lang.String pNLSLang)
```

This method is restricted.

getNLSLang(String)

```
public java.lang.String getNLSLang(java.lang.String pLangCode)
```

This method is restricted.

getSessionManager()

```
public oracle.apps.fnd.security.SessionManager getSessionManager()
```

This method is restricted.

getUrl()

```
public java.lang.String getUrl()
```

Returns the JDBC URL.

getUser()

```
public java.lang.String getUser()
```

Returns the user name.

releaseJDBCConnection()

```
public void releaseJDBCConnection()
```

setCurrLang(String)

```
public boolean setCurrLang(java.lang.String pLangCode)
```

Sets the current language code.

Package oracle.apps.cz.utilities

Description

Class Summary

Classes

NameValuePair	Provides a name-value pair object combination.
NameValuePairSet	Implements an object to hold a unique set of name value pairs

oracle.apps.cz.utilities NameValuePair

Syntax

```
public class NameValuePair extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.utilities.NameValuePair
```

Description

Provides a name-value pair object combination. The name cannot be changed once created.

Member Summary

Constructors

NameValuePair(String)	Constructs a name-value pair object without a value.
NameValuePair(String, Object)	Constructs a name-value pair object.

Methods

getName()	Retrieve the name (key).
getValue()	Retrieve the value which can be null.
setValue(Object)	Replaces the value for this pair.

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

NameValuePair(String)

```
public NameValuePair(java.lang.String key)
```

Constructs a name-value pair object without a value. Key is always stored as a lowercase string, regardless of the case of the key parameter.

Parameters:

key - - The String name(key) for this pair.

NameValuePair(String, Object)

```
public NameValuePair(java.lang.String key, java.lang.Object value)
```

Constructs a name-value pair object. Key is always stored as a lowercase string, regardless of the case of the key parameter.

Parameters:

key - - The String name(key) for this pair. String cannot be null or have only whitespace.

value - - The Object for this pair.

Methods

getName()

```
public java.lang.String getName()
```

Retrieve the name (key). Will always be lowercase.

getValue()

```
public java.lang.Object getValue()
```

Retrieve the value which can be null.

setValue(Object)

```
public void setValue(java.lang.Object value)
```

Replaces the value for this pair.

Parameters:

The - Object for this pair which can be null.

oracle.apps.cz.utilities NameValuePairSet

Syntax

```
public class NameValuePairSet extends java.lang.Object

java.lang.Object
|
+--oracle.apps.cz.utilities.NameValuePairSet
```

Description

Implements an object to hold a unique set of name value pairs

Member Summary

Constructors

[NameValuePairSet\(\)](#)

Methods

Add(NameValuePair)	Add a name value pair object to the set
Add(String, Object)	Create a NameValuePair and add it to the set using the name and object.
getValueByName(String)	Gets value, which may be null, of the name/value pair identified by the "name" input.
iterator()	Returns the set of keys for the name value pairs
lookupPairByName(String)	Look up a name (key) in the set

Inherited Member Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

NameValuePairSet()

```
public NameValuePairSet()
```

Methods

Add(NameValuePair)

```
public void Add(NameValuePair nvp)  
Add a name value pair object to the set
```

Parameters:

nameValuePair -- The NameValuePair object to add. The key will provide the hash.

Add(String, Object)

```
public void Add(java.lang.String name, java.lang.Object value)  
Create a NameValuePair and add it to the set using the name and object.
```

Parameters:

name -- The String key of the pair. The name will provide the hash identifier.

value -- The value object which can be null.

getValueByName(String)

```
public java.lang.Object getValueByName(java.lang.String name)  
Gets value, which may be null, of the name/value pair identified by the "name"  
input. Returns null if pair does not exist.
```

Parameters:

name - the name string by which to look up the value

Returns:

the value associated with name

iterator()

```
public java.util.Iterator iterator()
```

Returns the set of keys for the name value pairs

lookupPairByName(String)

```
public NameValuePair lookupPairByName(java.lang.String name)
```

Look up a name (key) in the set

Parameters:

name -- The String to lookup

Returns:

The NameValuePair which can be null if the string is not in the set

Index

A

- add() -
 - oracle.apps.cz.cio.ComponentSet.add(), B-58
- Add(NameValuePair) -
 - oracle.apps.cz.utilities.NameValuePairSet.Add(
oracle.apps.cz.utilities.NameValuePair), D-5
- Add(String, Object) -
 - oracle.apps.cz.utilities.NameValuePairSet.Add(
java.lang.String, java.lang.Object), D-5
- addAtpNotification(String) -
 - oracle.apps.cz.cio.BomNode.addAtpNotificatio
n(java.lang.String), B-17
- addConfigEventListener(ICompSetEventListener) -
 - oracle.apps.cz.cio.ComponentSet.addConfigEve
ntListener(oracle.apps.cz.cio.ICompSetEventLis
tener), B-58
- addConfigEventListener(IConfigEventListener) -
 - oracle.apps.cz.cio.Configuration.addConfigEve
ntListener(oracle.apps.cz.cio.IConfigEventLis
tner), B-65
- addConfigMessage(String, String) -
 - oracle.apps.cz.cio.Configuration.addConfigMes
sage(java.lang.String, java.lang.String), B-66
- addPricingNotification(String) -
 - oracle.apps.cz.cio.StateCountNode.addPricingN
otification(java.lang.String), B-211
- ALL_FEATURES -
 - oracle.apps.cz.cio.IRuntimeNode.ALL_
FEATURES, B-142
- areOptionsCounted() -
 - oracle.apps.cz.cio.BomNode.areOptionsCounte
d(), B-17
- areOptionsCounted() -
 - oracle.apps.cz.cio.IOptionFeature.areOptionsCo
unted(), B-133
- areOptionsCounted() -
 - oracle.apps.cz.cio.OptionFeature.areOptionsCo
unted(), B-175
- ATP_APPS_107_110 -
 - oracle.apps.cz.cio.Configuration.ATP_APPS_
107_110, B-65
- ATP_CALLBACK -
 - oracle.apps.cz.cio.Configuration.ATP_
CALLBACK, B-65
- ATP_DISABLED -
 - oracle.apps.cz.cio.Configuration.ATP_
DISABLED, B-65
- AtpUnavailableException -
 - oracle.apps.cz.cio.AtpUnavailableException, B
-7
- AtpUnavailableException(String) -
 - oracle.apps.cz.cio.AtpUnavailableException.At
pUnavailableException(java.lang.String), B-8
- AtpUnavailableException(String, Object, Log) -
 - oracle.apps.cz.cio.AtpUnavailableException.At
pUnavailableException(java.lang.String,
java.lang.Object,
oracle.apps.fnd.common.Log), B-8
- autoCommit
 - usage, 2-9
- Auto-configuration, 1-2, 1-11, 1-15, 2-9
 - exceptions, 2-19, 2-26
- autoConfigure()
 - usage, 2-9, 2-25
- autoConfigure() -
 - oracle.apps.cz.cio.FunctionalCompanion.autoC
onfigure(), B-97

autoConfigure() -
 oracle.apps.cz.cio.IFunctionalCompanion.autoConfigure(), B-116

AutoFunctionalCompanion -
 oracle.apps.cz.cio.AutoFunctionalCompanion, B-5

AutoFunctionalCompanion() -
 oracle.apps.cz.cio.AutoFunctionalCompanion.AutoFunctionalCompanion(), B-6

B

beginConfigTransaction()
 usage, 2-8

beginConfigTransaction() -
 oracle.apps.cz.cio.Configuration.beginConfigTransaction(), B-66

beginConfigTransaction(boolean) -
 oracle.apps.cz.cio.Configuration.beginConfigTransaction(boolean), B-66

BOM_MODEL -
 oracle.apps.cz.cio.IRuntimeNode.BOM_MODEL, B-142

BOM_OPTION_CLASS -
 oracle.apps.cz.cio.IRuntimeNode.BOM_OPTION_CLASS, B-142

BOM_STD_ITEM -
 oracle.apps.cz.cio.IRuntimeNode.BOM_STD_ITEM, B-142

BomExplosionException -
 oracle.apps.cz.cio.BomExplosionException, B-9

BomModel - oracle.apps.cz.cio.BomModel, B-11

BomNode - oracle.apps.cz.cio.BomNode, B-14

BomOptionClass -
 oracle.apps.cz.cio.BomOptionClass, B-26

BomStdItem - oracle.apps.cz.cio.BomStdItem, B-29

BOOLEAN_FEATURE -
 oracle.apps.cz.cio.IRuntimeNode.BOOLEAN_FEATURE, B-142

BooleanFeature -
 oracle.apps.cz.cio.BooleanFeature, B-32

C

calculateAtpDate() -
 oracle.apps.cz.cio.BomNode.calculateAtpDate(), B-17

calculateAtpDates() -
 oracle.apps.cz.cio.Configuration.calculateAtpDates(), B-66

calculateListPrices() -
 oracle.apps.cz.cio.Configuration.calculateListPrices(), B-67

canPerform() -
 oracle.apps.cz.cio.Configuration.canPerform(), B-67

canUndo() -
 oracle.apps.cz.cio.Configuration.canUndo(), B-67

CIO
 See Configuration Interface Object

CIO - oracle.apps.cz.cio.CIO, B-34

CIO() - oracle.apps.cz.cio.CIO.CIO(), B-35

classes111.zip, 1-8

CLASSPATH, 1-8

clearAtpDate() -
 oracle.apps.cz.cio.BomNode.clearAtpDate(), B-18

clearAtpNotifications() -
 oracle.apps.cz.cio.BomNode.clearAtpNotifications(), B-18

clearConfigMessages() -
 oracle.apps.cz.cio.Configuration.clearConfigMessages(), B-67

clearDiscountedPrice() -
 oracle.apps.cz.cio.StateCountNode.clearDiscountedPrice(), B-211

clearLogicFile(Object) -
 oracle.apps.cz.cio.CIO.clearLogicFile(java.lang.Object), B-35

clearLogicFileCache() -
 oracle.apps.cz.cio.CIO.clearLogicFileCache(), B-35

clearPricingNotifications() -
 oracle.apps.cz.cio.StateCountNode.clearPricingNotifications(), B-211

close() - oracle.apps.cz.cio.CIO.close(), B-36

- close() -
 - oracle.apps.cz.cio.Configuration.close(), B-67
- closeConfiguration()
 - usage, 2-6
- closeConfiguration(Configuration) -
 - oracle.apps.cz.cio.CIO.closeConfiguration(oracle.apps.cz.cio.Configuration), B-36
- closeTraceFile() -
 - oracle.apps.cz.cio.CIO.closeTraceFile(), B-36
- commitConfigTransaction(ConfigTransaction) -
 - oracle.apps.cz.cio.Configuration.commitConfigTransaction(oracle.apps.cz.cio.ConfigTransaction), B-67
- commitConfigTransaction(transaction)
 - usage, 2-8, 2-20, 2-21
- companion
 - See Functional Companion
- COMPANION_FAILURE -
 - oracle.apps.cz.cio.ValidationFailure.COMPANION_FAILURE, B-230
- CompanionNode -
 - oracle.apps.cz.cio.CompanionNode, B-42
- CompanionRoot -
 - oracle.apps.cz.cio.CompanionRoot, B-45
- CompanionRoot(IRuntimeNode) -
 - oracle.apps.cz.cio.CompanionRoot.CompanionRoot(oracle.apps.cz.cio.IRuntimeNode), B-46
- CompanionValidationFailure -
 - oracle.apps.cz.cio.CompanionValidationFailure, B-48
- CompanionValidationFailure(String, IRuntimeNode, IFunctionalCompanion) -
 - oracle.apps.cz.cio.CompanionValidationFailure.CompanionValidationFailure(java.lang.String, oracle.apps.cz.cio.IRuntimeNode, oracle.apps.cz.cio.IFunctionalCompanion), B-49
- Component - oracle.apps.cz.cio.Component, B-50
- COMPONENT -
 - oracle.apps.cz.cio.IRuntimeNode.COMPONENT, B-142
- COMPONENT_SET -
 - oracle.apps.cz.cio.IRuntimeNode.COMPONENT_SET, B-142
- ComponentNode -
 - oracle.apps.cz.cio.ComponentNode, B-55
- ComponentSet -
 - oracle.apps.cz.cio.ComponentSet, B-57
- ComponentSet.add()
 - usage, 2-10
- ComponentSet.delete(component)
 - usage, 2-10
- configHeaderID
 - usage, 2-7
- config.jar, 1-8
- ConfigTransaction -
 - oracle.apps.cz.cio.ConfigTransaction, B-60
- Configuration -
 - oracle.apps.cz.cio.Configuration, B-61
- configuration inputs
 - obtaining list of, 2-7
- Configuration Interface Object, 1-1
- configuration subschema objects, 2-1
- configuration-level logic transactions, 2-8
- configurations, 2-5
- constants, 3-1
- contradictions, 2-19, 2-26
- COUNT_FEATURE -
 - oracle.apps.cz.cio.IRuntimeNode.COUNT_FEATURE, B-142
- CountFeature -
 - oracle.apps.cz.cio.CountFeature, B-79
- createCIO() -
 - oracle.apps.cz.cio.Factory.createCIO(), B-90
- createConfiguration()
 - usage, 2-6
- createConfiguration(int, Context) -
 - oracle.apps.cz.cio.CIO.createConfiguration(int, oracle.apps.fnd.common.Context), B-36
- createConfiguration(int, int, Context) -
 - oracle.apps.cz.cio.CIO.createConfiguration(int, int, oracle.apps.fnd.common.Context), B-37
- createConfiguration(int, int, Date, Context) -
 - oracle.apps.cz.cio.CIO.createConfiguration(int, int, java.util.Date, oracle.apps.fnd.common.Context), B-37
- createConfiguration(String, Context) -
 - oracle.apps.cz.cio.CIO.createConfiguration(java.lang.String, oracle.apps.fnd.common.Context), B-38

- createConnection(String, String, String) -
 oracle.apps.cz.common.CZContext.createConne
 ction(java.lang.String, java.lang.String,
 java.lang.String), C-4
- createContext(String, String, String, String) -
 oracle.apps.cz.cio.Factory.createContext(java.la
 ng.String, java.lang.String, java.lang.String,
 java.lang.String), B-90
- createContext(String, String, String, String, String,
 String, String) -
 oracle.apps.cz.cio.Factory.createContext(java.la
 ng.String, java.lang.String, java.lang.String,
 java.lang.String, java.lang.String,
 java.lang.String, java.lang.String), B-90
- cz3rdpty.jar, 1-8, 1-10, 4-8
- CZContext -
 oracle.apps.cz.common.CZContext, C-2
- CZContext(String, String) -
 oracle.apps.cz.common.CZContext.CZContext(j
 ava.lang.String, java.lang.String), C-3
- CZContext(String, String, String, String) -
 oracle.apps.cz.common.CZContext.CZContext(j
 ava.lang.String, java.lang.String,
 java.lang.String, java.lang.String), C-3
- cz.dll, 1-8
- czjni.dll, 1-8

D

- DECIMAL_FEATURE -
 oracle.apps.cz.cio.IRuntimeNode.DECIMAL_
 FEATURE, B-143
- DecimalFeature -
 oracle.apps.cz.cio.DecimalFeature, B-83
- DecimalNode -
 oracle.apps.cz.cio.DecimalNode, B-86
- DEFAULT -
 oracle.apps.cz.cio.Reason.DEFAULT, B-192
- delete(Component) -
 oracle.apps.cz.cio.ComponentSet.delete(oracle.a
 pps.cz.cio.Component), B-59
- deselect()
 usage, 2-15
- deselect() -
 oracle.apps.cz.cio.BomNode.deselect(), B-18

- deselect() -
 oracle.apps.cz.cio.IOption.deselect(), B-131
- deselect() -
 oracle.apps.cz.cio.OptionNode.deselect(), B-1
 82
- deselect(IOption) -
 oracle.apps.cz.cio.IOptionFeature.deselect(orac
 le.apps.cz.cio.IOption), B-133
- deselect(IOption) -
 oracle.apps.cz.cio.OptionFeatureNode.deselect(
 oracle.apps.cz.cio.IOption), B-178
- drivers
 JDBC, 1-8

E

- endConfigTransaction(ConfigTransaction) -
 oracle.apps.cz.cio.Configuration.endConfigTran
 saction(oracle.apps.cz.cio.ConfigTransaction),
 B-67
- endConfigTransaction(transaction)
 usage, 2-8
- endDeltaList() -
 oracle.apps.cz.cio.Configuration.endDeltaList(),
 B-68
- equals(Object) -
 oracle.apps.cz.cio.CompanionValidationFailure.
 equals(java.lang.Object), B-49
- equals(Object) -
 oracle.apps.cz.cio.LogicalOverridableException.
 equals(java.lang.Object), B-157
- equals(Object) -
 oracle.apps.cz.cio.RestoreValidationFailure.equ
 als(java.lang.Object), B-198
- equals(Object) -
 oracle.apps.cz.cio.StatusInfo.equals(java.lang.O
 bject), B-219
- equals(Object) -
 oracle.apps.cz.cio.ValidationFailure.equals(java.
 lang.Object), B-231
- exceptions
 fatal, 2-19, 2-26
 logic, 2-18

F

Factory - oracle.apps.cz.cio.Factory, B-89
Factory() -
 oracle.apps.cz.cio.Factory.Factory(), B-90
FALSE
 usage, 2-11
FALSE - oracle.apps.cz.cio.IState.FALSE, B-149
FC_ON_AUTO_CONFIGURE -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_AUTO_CONFIGURE, B-115
FC_ON_GENERATE_OUTPUT -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_GENERATE_OUTPUT, B-115
FC_ON_NEW -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_NEW, B-115
FC_ON_RESTORE -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_RESTORE, B-115
FC_ON_SAVE -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_SAVE, B-115
FC_ON_VALIDATE -
 oracle.apps.cz.cio.IFunctionalCompanion.FC_ON_VALIDATE, B-115
finalizeWorkaround() -
 oracle.apps.cz.cio.Configuration.finalizeWorkaround(), B-68
FuncCompCreationException -
 oracle.apps.cz.cio.FuncCompCreationException, B-92
FuncCompMessageException -
 oracle.apps.cz.cio.FuncCompMessageException, B-94
FuncCompMessageException(IFunctionalCompanion, String) -
 oracle.apps.cz.cio.FuncCompMessageException.FuncCompMessageException(oracle.apps.cz.cio.IFunctionalCompanion, java.lang.String), B-95
Functional Companions
 and Project Structure, 1-10
 defined, 1-1
 relationship to CIO, 1-3, 2-2

 types, 1-1, 1-2
FunctionalCompanion -
 oracle.apps.cz.cio.FunctionalCompanion, B-96
FunctionalCompanion() -
 oracle.apps.cz.cio.FunctionalCompanion.FunctionalCompanion(), B-97
FunctionalCompanionException -
 oracle.apps.cz.cio.FunctionalCompanionException, B-100
FunctionalCompanionException(Throwable) -
 oracle.apps.cz.cio.FunctionalCompanionException.FunctionalCompanionException(java.lang.Throwable), B-100
FunctionalCompanionException(Throwable, String) -
 oracle.apps.cz.cio.FunctionalCompanionException.FunctionalCompanionException(java.lang.Throwable, java.lang.String), B-100

G

generateOutput()
 usage, 2-27
generateOutput() -
 oracle.apps.cz.cio.FunctionalCompanion.generateOutput(), B-97
generateOutput() -
 oracle.apps.cz.cio.IFunctionalCompanion.generateOutput(), B-116
generateOutput(HttpServletResponse) -
 oracle.apps.cz.cio.FunctionalCompanion.generateOutput(javax.servlet.http.HttpServletResponse), B-98
generateOutput(HttpServletResponse) -
 oracle.apps.cz.cio.IFunctionalCompanion.generateOutput(javax.servlet.http.HttpServletResponse), B-116
getAction() -
 oracle.apps.cz.cio.TransactionException.getAction(), B-228
getActiveModelPath() -
 oracle.apps.cz.cio.CIO.getActiveModelPath(), B-39
getAltPricingAtpContext() -
 oracle.apps.cz.cio.Configuration.getAltPricingAtpContext(), B-39

tpContext(), B-68
 getAppId(String) -
 oracle.apps.cz.common.CZContext.getAppId(java.lang.String), C-4
 getAtpDate() -
 oracle.apps.cz.cio.BomNode.getAtpDate(), B-18
 getAtpDate() -
 oracle.apps.cz.cio.IAtp.getAtpDate(), B-102
 getAtpMode() -
 oracle.apps.cz.cio.Configuration.getAtpMode(), B-68
 getAtpNotifications() -
 oracle.apps.cz.cio.BomNode.getAtpNotifications(), B-18
 getAtpNotifications() -
 oracle.apps.cz.cio.IAtp.getAtpNotifications(), B-102
 getBoolean(String) -
 oracle.apps.cz.cio.CompanionNode.getBoolean(java.lang.String), B-43
 getBoolean(String, boolean) -
 oracle.apps.cz.cio.CompanionNode.getBoolean(java.lang.String, boolean), B-43
 getBooleanValue() -
 oracle.apps.cz.cio.Property.getBooleanValue(), B-187
 getCause() -
 usage, 2-18
 getCause() -
 oracle.apps.cz.cio.LogicalException.getCause(), B-155
 getChildById(int) -
 oracle.apps.cz.cio.IRuntimeNode.getChildById(int), B-143
 getChildById(int) -
 oracle.apps.cz.cio.RuntimeNode.getChildById(int), B-201
 getChildByInstanceNumber(int) -
 oracle.apps.cz.cio.ComponentSet.getChildByInstanceNumber(int), B-59
 getChildByName(String) -
 oracle.apps.cz.cio.IRuntimeNode.getChildByName(java.lang.String), B-144
 getChildByName(String) -

 oracle.apps.cz.cio.RuntimeNode.getChildByName(java.lang.String), B-201
 getChildren() -
 oracle.apps.cz.cio.CompanionNode.getChildren(), B-43
 getChildren() -
 oracle.apps.cz.cio.Component.getChildren(), B-52
 getChildren() -
 oracle.apps.cz.cio.IRuntimeNode.getChildren(), B-144
 getChildren() -
 oracle.apps.cz.cio.RuntimeNode.getChildren(), B-201
 getChildrenByType() -
 usage, 2-17
 getChildrenByType(int) -
 oracle.apps.cz.cio.ComponentNode.getChildrenByType(int), B-56
 getChildrenByType(int) -
 oracle.apps.cz.cio.IRuntimeNode.getChildrenByType(int), B-144
 getChildrenByType(int) -
 oracle.apps.cz.cio.RuntimeNode.getChildrenByType(int), B-201
 getCIO() -
 usage, 2-6
 getCIO() -
 oracle.apps.cz.cio.Configuration.getCIO(), B-68
 getCompanion() -
 oracle.apps.cz.cio.CompanionValidationFailure.getCompanion(), B-49
 getComponentCode() -
 oracle.apps.cz.cio.BomNode.getComponentCode(), B-18
 getComponentCode() -
 oracle.apps.cz.cio.IBomItem.getComponentCode(), B-104
 getConfigHeaderCheckoutUser() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderCheckoutUser(), B-68
 getConfigHeaderDateCreated() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderDateCreated(), B-69

getConfigHeaderDescription() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderDescription(), B-69
 getConfigHeaderEffectiveFrom() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderEffectiveFrom(), B-69
 getConfigHeaderEffectiveTo() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderEffectiveTo(), B-69
 getConfigHeaderId() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderId(), B-69
 getConfigHeaderLastUpdateDate() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderLastUpdateDate(), B-70
 getConfigHeaderName() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderName(), B-70
 getConfigHeaderNote() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderNote(), B-70
 getConfigHeaderNumberQuotesUsedIn() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderNumberQuotesUsedIn(), B-70
 getConfigHeaderOpportunityHeaderId() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderOpportunityHeaderId(), B-70
 getConfigHeaderRevision() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderRevision(), B-71
 getConfigHeaderStatus() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderStatus(), B-71
 getConfigHeaderUiDefinitionId() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderUiDefinitionId(), B-71
 getConfigHeaderUserIdCreated() -
 oracle.apps.cz.cio.Configuration.getConfigHeaderUserIdCreated(), B-71
 getConfiguration()
 usage, 2-9
 getConfiguration() -
 oracle.apps.cz.cio.IRuntimeNode.getConfiguration(), B-144
 getConfiguration() -
 oracle.apps.cz.cio.RuntimeNode.getConfiguration(), B-202
 getContext() -
 oracle.apps.cz.cio.Configuration.getContext(), B-71
 getCount() -
 oracle.apps.cz.cio.Component.getCount(), B-52
 getCount() -
 oracle.apps.cz.cio.ComponentSet.getCount(), B-59
 getCount() -
 oracle.apps.cz.cio.ICount.getCount(), B-108
 getCount() -
 oracle.apps.cz.cio.StateCountNode.getCount(), B-211
 getCurrLangCode() -
 oracle.apps.cz.common.CZContext.getCurrLangCode(), C-4
 getCurrLangInfo() -
 oracle.apps.cz.common.CZContext.getCurrLangInfo(), C-4
 getDatabaseID() -
 oracle.apps.cz.cio.IAtp.getDatabaseID(), B-102
 getDatabaseID() -
 oracle.apps.cz.cio.IPrice.getDatabaseID(), B-136
 getDatabaseID() -
 oracle.apps.cz.cio.IRuntimeNode.getDatabaseID(), B-144
 getDatabaseID() -
 oracle.apps.cz.cio.RuntimeNode.getDatabaseID(), B-202
 getDbOwner() -
 oracle.apps.cz.common.CZContext.getDbOwner(), C-4
 getDecimalValue()
 usage, 2-13
 getDecimalValue() -
 oracle.apps.cz.cio.IReadOnlyDecimal.getDecimalValue(), B-138
 getDecimalValue() -
 oracle.apps.cz.cio.Property.getDecimalValue(), B-187
 getDecimalValue() -

- oracle.apps.cz.cio.ReadOnlyDecimalNode.getId
ecimalValue(), B-190
- getDefaultQuantity() -
oracle.apps.cz.cio.BomNode.getDefaultQuantit
y(), B-18
- getDescription() -
oracle.apps.cz.cio.FunctionalCompanion.getDes
cription(), B-98
- getDescription() -
oracle.apps.cz.cio.IFunctionalCompanion.getDe
scription(), B-116
- getDescription() -
oracle.apps.cz.cio.IRuntimeNode.getDescriptio
n(), B-145
- getDescription() -
oracle.apps.cz.cio.Property.getDescription(), B
-187
- getDescription() -
oracle.apps.cz.cio.RuntimeNode.getDescription
(), B-202
- getDescription() -
oracle.apps.cz.cio.FuncCompCreationException
.getDescription(), B-93
- getDiscountedPrice() -
oracle.apps.cz.cio.BomNode.getDiscountedPric
e(), B-18
- getDiscountedPrice() -
oracle.apps.cz.cio.IPrice.getDiscountedPrice(),
B-136
- getDiscountedPrice() -
oracle.apps.cz.cio.StateCountNode.getDiscount
edPrice(), B-211
- getDouble(String) -
oracle.apps.cz.cio.CompanionNode.getDouble(j
ava.lang.String), B-43
- getDouble(String, double) -
oracle.apps.cz.cio.CompanionNode.getDouble(j
ava.lang.String, double), B-43
- getEnvStore() -
oracle.apps.cz.common.CZContext.getEnvStore
(), C-4
- getExplosionDate() -
oracle.apps.cz.cio.BomExplosionException.getE
xplosionDate(), B-10
- getExtendedPrice() -

- oracle.apps.cz.cio.IPrice.getExtendedPrice(), B
-136
- getExtendedPrice() -
oracle.apps.cz.cio.StateCountNode.getExtended
Price(), B-212
- getFeature() -
oracle.apps.cz.cio.SelectionNotMutexedExcepti
on.getFeature(), B-208
- getFeature(String) -
oracle.apps.cz.cio.CompanionNode.getFeature(j
ava.lang.String), B-43
- getFeatureIdentifier() -
oracle.apps.cz.cio.CompanionRoot.getFeatureId
entifier(), B-46
- getFndMessageString(Message) -
oracle.apps.cz.cio.Reason.getFndMessageString
(oracle.apps.fnd.common.Message), B-194
- getFuncCompByID(int) -
oracle.apps.cz.cio.Component.getFuncCompByI
D(int), B-52
- getFuncCompByName(String) -
oracle.apps.cz.cio.Component.getFuncCompBy
Name(java.lang.String), B-52
- getFunctionalCompanions()
usage, 2-10
- getFunctionalCompanions() -
oracle.apps.cz.cio.Component.getFunctionalCo
mpanions(), B-52
- getID() -
oracle.apps.cz.cio.FuncCompCreationException
.getID(), B-93
- getID() -
oracle.apps.cz.cio.FunctionalCompanion.getID()
, B-98
- getID() -
oracle.apps.cz.cio.IFunctionalCompanion.getID(
) , B-116
- getID() -
oracle.apps.cz.cio.NoSuchChildException.getID
(), B-167
- getInitParameters()
usage, 2-7
- getInitParameters() -
oracle.apps.cz.cio.Configuration.getInitParamet
ers(), B-71

getInput() -
 oracle.apps.cz.cio.IncompatibleInputException.
 getInput(), B-123
 getInput() -
 oracle.apps.cz.cio.RestoreValidationFailure.getI
 nput(), B-198
 getInstanceNumber() -
 oracle.apps.cz.cio.Component.getInstanceNum
 ber(), B-52
 getInteger(String) -
 oracle.apps.cz.cio.CompanionNode.getInteger(j
 ava.lang.String), B-43
 getInteger(String, int) -
 oracle.apps.cz.cio.CompanionNode.getInteger(j
 ava.lang.String, int), B-44
 getIntValue()
 usage, 2-13
 getIntValue() -
 oracle.apps.cz.cio.CountFeature.getIntValue(),
 B-81
 getIntValue() -
 oracle.apps.cz.cio.IInteger.getIntValue(), B-11
 8
 getIntValue() -
 oracle.apps.cz.cio.IntegerNode.getIntValue(),
 B-128
 getIntValue() -
 oracle.apps.cz.cio.Property.getIntValue(), B-1
 87
 getInventoryItemId() -
 oracle.apps.cz.cio.BomExplosionException.getI
 nventoryItemId(), B-10
 getInventoryItemId() -
 oracle.apps.cz.cio.BomNode.getInventoryItemI
 d(), B-19
 getInventoryItemId() -
 oracle.apps.cz.cio.IBomItem.getInventoryItemI
 d(), B-104
 getItemKey() -
 oracle.apps.cz.cio.BomNode.getItemKey(), B-
 19
 getItemKey() -
 oracle.apps.cz.cio.IAtp.getItemKey(), B-102
 getItemKey() -
 oracle.apps.cz.cio.IPrice.getItemKey(), B-136
 getItemKey() -
 oracle.apps.cz.cio.StateCountNode.getItemKey(
), B-212
 getJDBCCConnection() -
 oracle.apps.cz.common.CZContext.getJDBCCon
 nection(), C-4
 getJDBCCConnection(Object) -
 oracle.apps.cz.common.CZContext.getJDBCCon
 nection(java.lang.Object), C-4
 getLangCode(String) -
 oracle.apps.cz.common.CZContext.getLangCod
 e(java.lang.String), C-5
 getLangInfo(String, String) -
 oracle.apps.cz.common.CZContext.getLangInfo
 (java.lang.String, java.lang.String), C-5
 getLastContradiction() -
 oracle.apps.cz.cio.Configuration.getLastContra
 diction(), B-72
 getListPrice() -
 oracle.apps.cz.cio.BomNode.getListPrice(), B-
 19
 getListPrice() -
 oracle.apps.cz.cio.IPrice.getListPrice(), B-136
 getListPrice() -
 oracle.apps.cz.cio.StateCountNode.getListPrice(
), B-212
 getLogicConfig() -
 oracle.apps.cz.cio.Configuration.getLogicConfig
 (), B-72
 getLogicException() -
 oracle.apps.cz.cio.LogicalRuntimeException.get
 LogicException(), B-159
 getMax() -
 oracle.apps.cz.cio.Component.getMax(), B-52
 getMax() -
 oracle.apps.cz.cio.ComponentSet.getMax(), B-
 59
 getMax() -
 oracle.apps.cz.cio.CountFeature.getMax(), B-8
 1
 getMax() -
 oracle.apps.cz.cio.DecimalFeature.getMax(), B
 -84
 getMax() -
 oracle.apps.cz.cio.IDecimalMinMax.getMax(),

B-112
 getMax() -
 oracle.apps.cz.cio.IIntegerMinMax.getMax(),
 B-120
 getMax() -
 oracle.apps.cz.cio.IntegerFeature.getMax(), B-
 125
 getMaxQuantity() -
 oracle.apps.cz.cio.BomNode.getMaxQuantity(),
 B-20
 getMaxQuantity() -
 oracle.apps.cz.cio.IBomItem.getMaxQuantity(),
 B-104
 getMaxSelected()
 usage, 2-11
 getMaxSelected() -
 oracle.apps.cz.cio.BomNode.getMaxSelected(),
 B-20
 getMaxSelected() -
 oracle.apps.cz.cio.IOptionFeature.getMaxSelect
 ed(), B-133
 getMaxSelected() -
 oracle.apps.cz.cio.OptionFeature.getMaxSelecte
 d(), B-175
 getMessage()
 usage, 2-18
 getMessage() -
 oracle.apps.cz.cio.LogicalException.getMessage
 (), B-155
 getMessage() -
 oracle.apps.cz.cio.ValidationFailure.getMessage
 (), B-231
 getMessage(String) -
 oracle.apps.cz.cio.ValidationFailure.getMessage
 (java.lang.String), B-231
 getMessageHeader() -
 oracle.apps.cz.cio.LogicalException.getMessage
 Header(), B-155
 getMin() -
 oracle.apps.cz.cio.Component.getMin(), B-52
 getMin() -
 oracle.apps.cz.cio.ComponentSet.getMin(), B-
 59
 getMin() -
 oracle.apps.cz.cio.CountFeature.getMin(), B-8

1
 getMin() -
 oracle.apps.cz.cio.DecimalFeature.getMin(), B
 -85
 getMin() -
 oracle.apps.cz.cio.IDecimalMinMax.getMin(),
 B-112
 getMin() -
 oracle.apps.cz.cio.IIntegerMinMax.getMin(), B
 -120
 getMin() -
 oracle.apps.cz.cio.IntegerFeature.getMin(), B-
 125
 getMinQuantity() -
 oracle.apps.cz.cio.BomNode.getMinQuantity(),
 B-20
 getMinQuantity() -
 oracle.apps.cz.cio.IBomItem.getMinQuantity(),
 B-105
 getMinSelected()
 usage, 2-11
 getMinSelected() -
 oracle.apps.cz.cio.BomNode.getMinSelected(),
 B-20
 getMinSelected() -
 oracle.apps.cz.cio.IOptionFeature.getMinSelecte
 d(), B-133
 getMinSelected() -
 oracle.apps.cz.cio.OptionFeature.getMinSelecte
 d(), B-175
 getModelNode() -
 oracle.apps.cz.cio.IncompatibleInputException.
 getModelNode(), B-123
 getMsg()
 usage, 2-19
 getMsg() -
 oracle.apps.cz.cio.Reason.getMsg(), B-194
 getName() -
 oracle.apps.cz.cio.Component.getName(), B-5
 3
 getName() -
 oracle.apps.cz.cio.FuncCompCreationException
 .getName(), B-93
 getName() -
 oracle.apps.cz.cio.FunctionalCompanion.getNa

me(), B-98
 getName() -
 oracle.apps.cz.cio.IFunctionalCompanion.getName(), B-116
 getName() -
 oracle.apps.cz.cio.IRuntimeNode.getName(), B-145
 getName() -
 oracle.apps.cz.cio.NoSuchChildException.getName(), B-167
 getName() -
 oracle.apps.cz.cio.Property.getName(), B-187
 getName() -
 oracle.apps.cz.cio.RuntimeNode.getName(), B-202
 getName() -
 oracle.apps.cz.utilities.NameValuePair.getName(), D-3
 getNLSLang(String) -
 oracle.apps.cz.common.CZContext.getNLSLang(java.lang.String), C-5
 getNode() -
 usage, 2-19
 getNode() -
 oracle.apps.cz.cio.Reason.getNode(), B-194
 getNode() -
 oracle.apps.cz.cio.StatusInfo.getNode(), B-219
 getNodeClass(String) -
 oracle.apps.cz.cio.CompanionRoot.getNodeClass(java.lang.String), B-46
 getNodeFromPath(String) -
 oracle.apps.cz.cio.Configuration.getNodeFromPath(java.lang.String), B-72
 getNodeIdentifier() -
 oracle.apps.cz.cio.CompanionRoot.getNodeIdentifier(), B-46
 getOrganizationId() -
 oracle.apps.cz.cio.BomExplosionException.getOrganizationId(), B-10
 getOrganizationId() -
 oracle.apps.cz.cio.BomNode.getOrganizationId(), B-21
 getOrganizationId() -
 oracle.apps.cz.cio.IBomItem.getOrganizationId(), B-105
 getParent() -
 oracle.apps.cz.cio.IRuntimeNode.getParent(), B-145
 getParent() -
 oracle.apps.cz.cio.NoSuchChildException.getParent(), B-167
 getParent() -
 oracle.apps.cz.cio.RuntimeNode.getParent(), B-202
 getPathForNode(IRuntimeNode) -
 oracle.apps.cz.cio.Configuration.getPathForNode(oracle.apps.cz.cio.IRuntimeNode), B-72
 getPricingMode() -
 oracle.apps.cz.cio.Configuration.getPricingMode(), B-72
 getPricingNotifications() -
 oracle.apps.cz.cio.IPrice.getPricingNotifications(), B-136
 getPricingNotifications() -
 oracle.apps.cz.cio.StateCountNode.getPricingNotifications(), B-212
 getPrimaryUomCode() -
 oracle.apps.cz.cio.BomNode.getPrimaryUomCode(), B-21
 getPrimaryUomCode() -
 oracle.apps.cz.cio.IBomItem.getPrimaryUomCode(), B-105
 getProgString() -
 oracle.apps.cz.cio.FuncCompCreationException.getProgString(), B-93
 getProjectID() -
 oracle.apps.cz.cio.Configuration.getProjectID(), B-72
 getProperties() -
 usage, 2-14
 getProperties() -
 oracle.apps.cz.cio.IRuntimeNode.getProperties(), B-145
 getProperties() -
 oracle.apps.cz.cio.RuntimeNode.getProperties(), B-203
 getPropertyByName() -
 usage, 2-14
 getPropertyByName(String) -
 oracle.apps.cz.cio.IRuntimeNode.getPropertyBy

- Name(java.lang.String), B-145
- getPropertyByName(String) -
 - oracle.apps.cz.cio.RuntimeNode.getPropertyByName(java.lang.String), B-203
- getReasons()
 - usage, 2-18
- getReasons() -
 - oracle.apps.cz.cio.LogicalException.getReasons(), B-155
- getRootBomModel() -
 - oracle.apps.cz.cio.Configuration.getRootBomModel(), B-72
- getRootBomModel(int, int) -
 - oracle.apps.cz.cio.Configuration.getRootBomModel(int, int), B-73
- getRootComponent() -
 - oracle.apps.cz.cio.Configuration.getRootComponent(), B-73
- getRootComponentDbId() -
 - oracle.apps.cz.cio.Configuration.getRootComponentDbId(), B-73
- getRootNodes() -
 - oracle.apps.cz.cio.CompanionRoot.getRootNodes(), B-47
- getRuntimeID() -
 - oracle.apps.cz.cio.IRuntimeNode.getRuntimeID(), B-145
- getRuntimeID() -
 - oracle.apps.cz.cio.RuntimeNode.getRuntimeID(), B-203
- getRuntimeNode() -
 - oracle.apps.cz.cio.FunctionalCompanion.getRuntimeNode(), B-98
- getRuntimeNode() -
 - oracle.apps.cz.cio.IFunctionalCompanion.getRuntimeNode(), B-116
- getRuntimeNode(int) -
 - oracle.apps.cz.cio.Configuration.getRuntimeNode(int), B-73
- getSelectedItems()
 - usage, 2-22
- getSelectedItems() -
 - oracle.apps.cz.cio.Configuration.getSelectedItems(), B-74
- getSelectedOption()

- usage, 2-14
- getSelectedOption() -
 - oracle.apps.cz.cio.IOptionFeature.getSelectedOption(), B-134
- getSelectedOption() -
 - oracle.apps.cz.cio.OptionFeatureNode.getSelectedOption(), B-178
- getSelectedOptions() -
 - oracle.apps.cz.cio.IOptionFeature.getSelectedOptions(), B-134
- getSelectedOptions() -
 - oracle.apps.cz.cio.OptionFeatureNode.getSelectedOptions(), B-178
- getSelectionLineID() -
 - oracle.apps.cz.cio.IRuntimeNode.getSelectionLineID(), B-146
- getSelectionLineID() -
 - oracle.apps.cz.cio.RuntimeNode.getSelectionLineID(), B-203
- getSessionManager() -
 - oracle.apps.cz.common.CZContext.getSessionManager(), C-5
- getState()
 - usage, 2-12
- getState() -
 - oracle.apps.cz.cio.BomNode.getState(), B-21
- getState() -
 - oracle.apps.cz.cio.IState.getState(), B-150
- getState() -
 - oracle.apps.cz.cio.StateNode.getState(), B-215
- getStatus() -
 - oracle.apps.cz.cio.StatusInfo.getStatus(), B-220
- getString(String) -
 - oracle.apps.cz.cio.CompanionNode.getString(java.lang.String), B-44
- getString(String, String) -
 - oracle.apps.cz.cio.CompanionNode.getString(java.lang.String, java.lang.String), B-44
- getStringDeltaList() -
 - oracle.apps.cz.cio.Configuration.getStringDeltaList(), B-74
- getStringValue()
 - usage, 2-14
- getStringValue() -
 - oracle.apps.cz.cio.Property.getStringValue(), B

- 187
- getTextValue() -
 - oracle.apps.cz.cio.IText.getTextValue(), B-151
- getTextValue() -
 - oracle.apps.cz.cio.TextNode.getTextValue(), B-224
- getTotalDiscountedPrice() -
 - oracle.apps.cz.cio.Configuration.getTotalDiscountedPrice(), B-74
- getTransactionDepth() -
 - oracle.apps.cz.cio.Configuration.getTransactionDepth(), B-74
- getType()
 - usage, 2-19
- getType() -
 - oracle.apps.cz.cio.BomModel.getType(), B-13
- getType() -
 - oracle.apps.cz.cio.BomOptionClass.getType(), B-28
- getType() -
 - oracle.apps.cz.cio.BomStdItem.getType(), B-31
- getType() -
 - oracle.apps.cz.cio.BooleanFeature.getType(), B-33
- getType() -
 - oracle.apps.cz.cio.Component.getType(), B-53
- getType() -
 - oracle.apps.cz.cio.ComponentSet.getType(), B-59
- getType() -
 - oracle.apps.cz.cio.CountFeature.getType(), B-81
- getType() -
 - oracle.apps.cz.cio.DecimalFeature.getType(), B-85
- getType() -
 - oracle.apps.cz.cio.IntegerFeature.getType(), B-126
- getType() -
 - oracle.apps.cz.cio.IRuntimeNode.getType(), B-146
- getType() -
 - oracle.apps.cz.cio.OptionFeature.getType(), B-175
- getType() -
 - oracle.apps.cz.cio.Option.getType(), B-172
- getType() -
 - oracle.apps.cz.cio.Reason.getType(), B-194
- getType() -
 - oracle.apps.cz.cio.Resource.getType(), B-196
- getType() -
 - oracle.apps.cz.cio.RuntimeNode.getType(), B-203
- getType() -
 - oracle.apps.cz.cio.TextFeature.getType(), B-222
- getType() -
 - oracle.apps.cz.cio.Total.getType(), B-226
- getType() -
 - oracle.apps.cz.cio.ValidationFailure.getType(), B-231
- getUnit() -
 - oracle.apps.cz.cio.Property.getUnit(), B-187
- getUnsatisfiedItems()
 - usage, 2-22
- getUnsatisfiedItems() -
 - oracle.apps.cz.cio.Configuration.getUnsatisfiedItems(), B-74
- getUomCode() -
 - oracle.apps.cz.cio.BomNode.getUomCode(), B-21
- getUomCode() -
 - oracle.apps.cz.cio.IAtp.getUomCode(), B-102
- getUomCode() -
 - oracle.apps.cz.cio.IPrice.getUomCode(), B-137
- getUomCode() -
 - oracle.apps.cz.cio.StateCountNode.getUomCode(), B-212
- getUrl() -
 - oracle.apps.cz.common.CZContext.getUrl(), C-5
- getUser() -
 - oracle.apps.cz.common.CZContext.getUser(), C-5
- getValidationFailures()
 - usage, 2-13, 2-22
- getValidationFailures() -
 - oracle.apps.cz.cio.Configuration.getValidationFailures(), B-74

getValue() -
 oracle.apps.cz.utilities.NameValuePair.getValue()
 (), D-3
getValueByName(String) -
 oracle.apps.cz.utilities.NameValuePairSet.getValueByName(java.lang.String), D-5

H

hasBooleanValue() -
 oracle.apps.cz.cio.Property.hasBooleanValue(),
 B-188
hasCount() -
 oracle.apps.cz.cio.IRuntimeNode.hasCount(),
 B-146
hasCount() -
 oracle.apps.cz.cio.RuntimeNode.hasCount(), B-204
hasDecimalValue() -
 oracle.apps.cz.cio.IRuntimeNode.hasDecimalValue(), B-146
hasDecimalValue() -
 oracle.apps.cz.cio.Property.hasDecimalValue(),
 B-188
hasDecimalValue() -
 oracle.apps.cz.cio.RuntimeNode.hasDecimalValue(), B-204
hasDefaultQuantity() -
 oracle.apps.cz.cio.BomNode.hasDefaultQuantity(), B-22
hasDefaultValue() -
 oracle.apps.cz.cio.Property.hasDefaultValue(),
 B-188
hasDescription() -
 oracle.apps.cz.cio.IRuntimeNode.hasDescription(), B-146
hasDescription() -
 oracle.apps.cz.cio.RuntimeNode.hasDescription(), B-204
hasFeature(String) -
 oracle.apps.cz.cio.CompanionNode.hasFeature(java.lang.String), B-44
hashCode() -
 oracle.apps.cz.cio.CompanionValidationFailure.hashCode(), B-49

hashCode() -
 oracle.apps.cz.cio.RestoreValidationFailure.hashCode(), B-198
hashCode() -
 oracle.apps.cz.cio.StatusInfo.hashCode(), B-220
hasIntegerValue() -
 oracle.apps.cz.cio.Property.hasIntegerValue(),
 B-188
hasIntegerValue() -
 oracle.apps.cz.cio.RuntimeNode.hasIntegerValue(), B-204
hasMax()
 usage, 2-11
hasMax() -
 oracle.apps.cz.cio.Component.hasMax(), B-53
hasMax() -
 oracle.apps.cz.cio.ComponentSet.hasMax(), B-59
hasMax() -
 oracle.apps.cz.cio.CountFeature.hasMax(), B-82
hasMax() -
 oracle.apps.cz.cio.DecimalFeature.hasMax(), B-85
hasMax() -
 oracle.apps.cz.cio.IDecimalMinMax.hasMax(),
 B-112
hasMax() -
 oracle.apps.cz.cio.IIntegerMinMax.hasMax(),
 B-120
hasMax() -
 oracle.apps.cz.cio.IntegerFeature.hasMax(), B-126
hasMaxQuantity() -
 oracle.apps.cz.cio.BomNode.hasMaxQuantity(),
 B-22
hasMaxQuantity() -
 oracle.apps.cz.cio.IBomItem.hasMaxQuantity(),
 B-105
hasMaxSelected()
 usage, 2-11
hasMaxSelected() -
 oracle.apps.cz.cio.BomNode.hasMaxSelected(),
 B-22

- hasMaxSelected() -
 - oracle.apps.cz.cio.IOptionFeature.hasMaxSelected(), B-134
- hasMaxSelected() -
 - oracle.apps.cz.cio.OptionFeature.hasMaxSelected(), B-175
- hasMin()
 - usage, 2-11
- hasMin() -
 - oracle.apps.cz.cio.Component.hasMin(), B-53
- hasMin() -
 - oracle.apps.cz.cio.ComponentSet.hasMin(), B-59
- hasMin() -
 - oracle.apps.cz.cio.CountFeature.hasMin(), B-82
- hasMin() -
 - oracle.apps.cz.cio.DecimalFeature.hasMin(), B-85
- hasMin() -
 - oracle.apps.cz.cio.IDecimalMinMax.hasMin(), B-113
- hasMin() -
 - oracle.apps.cz.cio.IIntegerMinMax.hasMin(), B-121
- hasMin() -
 - oracle.apps.cz.cio.IntegerFeature.hasMin(), B-126
- hasMinQuantity() -
 - oracle.apps.cz.cio.BomNode.hasMinQuantity(), B-22
- hasMinQuantity() -
 - oracle.apps.cz.cio.IBomItem.hasMinQuantity(), B-105
- hasMinSelected()
 - usage, 2-11
- hasMinSelected() -
 - oracle.apps.cz.cio.BomNode.hasMinSelected(), B-22
- hasMinSelected() -
 - oracle.apps.cz.cio.IOptionFeature.hasMinSelected(), B-134
- hasMinSelected() -
 - oracle.apps.cz.cio.OptionFeature.hasMinSelected(), B-175

- hasSelectionLineID() -
 - oracle.apps.cz.cio.IRuntimeNode.hasSelectionLineID(), B-146
- hasSelectionLineID() -
 - oracle.apps.cz.cio.RuntimeNode.hasSelectionLineID(), B-204
- hasState() -
 - oracle.apps.cz.cio.IRuntimeNode.hasState(), B-146
- hasState() -
 - oracle.apps.cz.cio.RuntimeNode.hasState(), B-204
- hasStringValue() -
 - oracle.apps.cz.cio.Property.hasStringValue(), B-188
- hasTextValue() -
 - oracle.apps.cz.cio.IRuntimeNode.hasTextValue(), B-147
- hasTextValue() -
 - oracle.apps.cz.cio.RuntimeNode.hasTextValue(), B-205

I

- IAtp - oracle.apps.cz.cio.IAtp, B-101
- IBomItem - oracle.apps.cz.cio.IBomItem, B-103
- ICompSetEventListener -
 - oracle.apps.cz.cio.ICompSetEventListener, B-106
- IConfigEventListener -
 - oracle.apps.cz.cio.IConfigEventListener, B-107
- ICount - oracle.apps.cz.cio.ICount, B-108
- IDecimal - oracle.apps.cz.cio.IDecimal, B-110
- IDecimalMinMax -
 - oracle.apps.cz.cio.IDecimalMinMax, B-112
- IFunctionalCompanion, 2-9, 2-10, 2-23, 2-25, 2-26, 2-28
- IFunctionalCompanion -
 - oracle.apps.cz.cio.IFunctionalCompanion, B-114
- IInteger - oracle.apps.cz.cio.IInteger, B-118
- IIntegerMinMax -
 - oracle.apps.cz.cio.IIntegerMinMax, B-120
- IncompatibleInputException -

- oracle.apps.cz.cio.IncompatibleInputException, B-122
- Initialization, 2-10
- initialization parameters
 - obtaining list of, 2-7
- initialize()
 - usage, 2-10, 2-23
- initialize(IRuntimeNode, String, String, int) -
 - oracle.apps.cz.cio.FunctionalCompanion.initialize(oracle.apps.cz.cio.IRuntimeNode, java.lang.String, java.lang.String, int), B-99
- initialize(IRuntimeNode, String, String, int) -
 - oracle.apps.cz.cio.IFunctionalCompanion.initialize(oracle.apps.cz.cio.IRuntimeNode, java.lang.String, java.lang.String, int), B-117
- initTraceFile(String) -
 - oracle.apps.cz.cio.CIO.initTraceFile(java.lang.String), B-39
- instanceTypeToString(int) -
 - oracle.apps.cz.cio.Component.instanceTypeToString(int), B-53
- INTEGER_FEATURE -
 - oracle.apps.cz.cio.IRuntimeNode.INTEGER_FEATURE, B-143
- IntegerFeature -
 - oracle.apps.cz.cio.IntegerFeature, B-124
- IntegerNode -
 - oracle.apps.cz.cio.IntegerNode, B-127
- interface
 - methods, 2-22
 - objects, 1-4, 2-2
- INTL_TEXT - oracle.apps.cz.cio.Reason.INTL_TEXT, B-192
- IOption - oracle.apps.cz.cio.IOption, B-130
- IOptionFeature -
 - oracle.apps.cz.cio.IOptionFeature, B-132
- IPrice - oracle.apps.cz.cio.IPrice, B-135
- IReadOnlyDecimal -
 - oracle.apps.cz.cio.IReadOnlyDecimal, B-138
- IRuntimeNode -
 - oracle.apps.cz.cio.IRuntimeNode, B-140
- isActive() -
 - oracle.apps.cz.cio.ComponentNode.isActive(), B-56
- isDefaultState(int) -

- oracle.apps.cz.cio.StateNode.isDefaultState(int), B-216
- isFalseState(int) -
 - oracle.apps.cz.cio.StateNode.isFalseState(int), B-216
- isLogicState(int) -
 - oracle.apps.cz.cio.StateNode.isLogicState(int), B-216
- isNative() -
 - oracle.apps.cz.cio.IRuntimeNode.isNative(), B-147
- isNative() -
 - oracle.apps.cz.cio.RuntimeNode.isNative(), B-205
- isOptionMutexed() -
 - oracle.apps.cz.cio.BomNode.isOptionMutexed(), B-23
- isOptionMutexed() -
 - oracle.apps.cz.cio.IOption.isOptionMutexed(), B-131
- isOptionMutexed() -
 - oracle.apps.cz.cio.OptionNode.isOptionMutexed(), B-182
- isOverridable()
 - usage, 2-18
- isOverridable() -
 - oracle.apps.cz.cio.LogicalException.isOverridable(), B-155
- isOverridable() -
 - oracle.apps.cz.cio.LogicalOverridableException.isOverridable(), B-157
- isRequired() -
 - oracle.apps.cz.cio.BomNode.isRequired(), B-23
- isRequired() -
 - oracle.apps.cz.cio.IBomItem.isRequired(), B-105
- isRoot() -
 - oracle.apps.cz.cio.Component.isRoot(), B-53
- isSelected()
 - usage, 2-15
- isSelected() -
 - oracle.apps.cz.cio.BomNode.isSelected(), B-23
- isSelected() -
 - oracle.apps.cz.cio.IOption.isSelected(), B-131

- isSelected() -
 - oracle.apps.cz.cio.OptionNode.isSelected(), B-182
- isSelectionMutexed() -
 - oracle.apps.cz.cio.BomNode.isSelectionMutexed(), B-23
- isSelectionMutexed() -
 - oracle.apps.cz.cio.IOptionFeature.isSelectionMutexed(), B-134
- isSelectionMutexed() -
 - oracle.apps.cz.cio.OptionFeatureNode.isSelectionMutexed(), B-178
- IState - oracle.apps.cz.cio.IState, B-148
- isTrueState(int) -
 - oracle.apps.cz.cio.StateNode.isTrueState(int), B-216
- isUnknown() -
 - oracle.apps.cz.cio.IInteger.isUnknown(), B-118
- isUnknown() -
 - oracle.apps.cz.cio.IntegerNode.isUnknown(), B-128
- isUnknown() -
 - oracle.apps.cz.cio.IReadOnlyDecimal.isUnknown(), B-138
- isUnknown() -
 - oracle.apps.cz.cio.IState.isUnknown(), B-150
- isUnknown() -
 - oracle.apps.cz.cio.ReadOnlyDecimalNode.isUnknown(), B-190
- isUnknown() -
 - oracle.apps.cz.cio.StateNode.isUnknown(), B-216
- isUnknownState(int) -
 - oracle.apps.cz.cio.StateNode.isUnknownState(int), B-216
- isUnsatisfied() -
 - oracle.apps.cz.cio.Configuration.isUnsatisfied(), B-75
- isUnsatisfied() -
 - oracle.apps.cz.cio.IRuntimeNode.isUnsatisfied(), B-147
- isUnsatisfied() -
 - oracle.apps.cz.cio.RuntimeNode.isUnsatisfied(), B-205

- isUnsatisfiedNode() -
 - oracle.apps.cz.cio.IRuntimeNode.isUnsatisfiedNode(), B-147
- isUnsatisfiedNode() -
 - oracle.apps.cz.cio.RuntimeNode.isUnsatisfiedNode(), B-205
- isUserState(int) -
 - oracle.apps.cz.cio.StateNode.isUserState(int), B-216
- isVirtual() -
 - oracle.apps.cz.cio.Component.isVirtual(), B-53
- iterator() -
 - oracle.apps.cz.utilities.NameValuePairSet.iterator(), D-5
- IText - oracle.apps.cz.cio.IText, B-151

J

- Java
 - building Functional Companions with, 1-3, 1-4
 - packages for the CIO, 2-1
 - recommended for development, 1-3
 - specifying Functional Companion type, 1-11
- JDBC
 - drivers, 1-8

L

- LFALSE
 - usage, 2-11
- LFALSE - oracle.apps.cz.cio.IState.LFALSE, B-149
- loadDriver(String) -
 - oracle.apps.cz.cio.Factory.loadDriver(java.lang.String), B-91
- logic
 - contradictions, 2-18
 - exceptions, 2-18
 - net objects, 2-1
 - transactions, 2-8, 2-21
- LogicalException -
 - oracle.apps.cz.cio.LogicalException, B-153
- LogicalException() -
 - oracle.apps.cz.cio.LogicalException.LogicalException(), B-154

LogicalException(LogicException, Configuration) -
 oracle.apps.cz.cio.LogicalException.LogicalException(oracle.apps.cz.logic.LogicException, oracle.apps.cz.cio.Configuration), B-154
 LogicalException(Reason, Configuration) -
 oracle.apps.cz.cio.LogicalException.LogicalException(oracle.apps.cz.cio.Reason, oracle.apps.cz.cio.Configuration), B-154
 LogicalException(String, Configuration) -
 oracle.apps.cz.cio.LogicalException.LogicalException(java.lang.String, oracle.apps.cz.cio.Configuration), B-154
 LogicalOverridableException -
 oracle.apps.cz.cio.LogicalOverridableException, B-156
 LogicalOverridableException(LogicContradictionException, Configuration) -
 oracle.apps.cz.cio.LogicalOverridableException.LogicalOverridableException(oracle.apps.cz.logic.LogicContradictionException, oracle.apps.cz.cio.Configuration), B-157
 LogicalRuntimeException -
 oracle.apps.cz.cio.LogicalRuntimeException, B-158
 LogicalRuntimeException(LogicException) -
 oracle.apps.cz.cio.LogicalRuntimeException.LogicalRuntimeException(oracle.apps.cz.logic.LogicException), B-159
 LogicalRuntimeException(String) -
 oracle.apps.cz.cio.LogicalRuntimeException.LogicalRuntimeException(java.lang.String), B-159
 lookupNodeID(Object) -
 oracle.apps.cz.cio.Component.lookupNodeID(java.lang.Object), B-53
 lookupNodeID(Object) -
 oracle.apps.cz.cio.RuntimeNode.lookupNodeID(java.lang.Object), B-205
 lookupPairByName(String) -
 oracle.apps.cz.utilities.NameValuePairSet.lookupPairByName(java.lang.String), D-6
 LTRUE
 usage, 2-11
 LTRUE - oracle.apps.cz.cio.IState.LTRUE, B-149

M

MAX_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.MAX_FAILURE, B-230
 MIN_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.MIN_FAILURE, B-230
 MIN0_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.MIN0_FAILURE, B-230
 MINMAX -
 oracle.apps.cz.cio.Reason.MINMAX, B-192
 MINMAX_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.MINMAX_FAILURE, B-230
 MissingFileException -
 oracle.apps.cz.cio.MissingFileException, B-160
 MissingLogicException -
 oracle.apps.cz.cio.MissingLogicException, B-161

N

NameValuePair -
 oracle.apps.cz.utilities.NameValuePair, D-2
 NameValuePair(String) -
 oracle.apps.cz.utilities.NameValuePair.NameValuePair(java.lang.String), D-3
 NameValuePair(String, Object) -
 oracle.apps.cz.utilities.NameValuePair.NameValuePair(java.lang.String, java.lang.Object), D-3
 NameValuePairSet -
 oracle.apps.cz.utilities.NameValuePairSet, D-4
 NameValuePairSet() -
 oracle.apps.cz.utilities.NameValuePairSet.NameValuePairSet(), D-5
 native BOM, 2-16
 New Functional Companion command, 1-11
 NoAtpCalculatedException -
 oracle.apps.cz.cio.NoAtpCalculatedException, B-162
 NoConfigHeaderException -

- oracle.apps.cz.cio.NoConfigHeaderException, B-163
- NoConfigHeaderException() -
 - oracle.apps.cz.cio.NoConfigHeaderException.NoConfigHeaderException(), B-164
- NonPricedNodeException -
 - oracle.apps.cz.cio.NonPricedNodeException, B-165
- NoSuchChildException -
 - oracle.apps.cz.cio.NoSuchChildException, B-166
- NoSuchChildException(IRuntimeNode, int) -
 - oracle.apps.cz.cio.NoSuchChildException.NoSuchChildException(oracle.apps.cz.cio.IRuntimeNode, int), B-167
- NoSuchChildException(IRuntimeNode, String) -
 - oracle.apps.cz.cio.NoSuchChildException.NoSuchChildException(oracle.apps.cz.cio.IRuntimeNode, java.lang.String), B-167
- notifyComponentAdded(Component) -
 - oracle.apps.cz.cio.ICompSetEventListener.notifyComponentAdded(oracle.apps.cz.cio.Component), B-106
- notifyComponentAdded(Component) -
 - oracle.apps.cz.cio.IConfigEventListener.notifyComponentAdded(oracle.apps.cz.cio.Component), B-107
- notifyComponentDeleted(Component) -
 - oracle.apps.cz.cio.ICompSetEventListener.notifyComponentDeleted(oracle.apps.cz.cio.Component), B-106
- notifyComponentDeleted(Component) -
 - oracle.apps.cz.cio.IConfigEventListener.notifyComponentDeleted(oracle.apps.cz.cio.Component), B-107
- NotOneProductException -
 - oracle.apps.cz.cio.NotOneProductException, B-168
- NotOneProjectException -
 - oracle.apps.cz.cio.NotOneProjectException, B-169
- oracle.apps.cz.cio.AutoFunctionalCompanion.onNew(), B-6
- onRestore() -
 - oracle.apps.cz.cio.AutoFunctionalCompanion.onRestore(), B-6
- onSave() -
 - oracle.apps.cz.cio.AutoFunctionalCompanion.onSave(), B-6
- openTraceFile(String) -
 - oracle.apps.cz.cio.CIO.openTraceFile(java.lang.String), B-39
- OPTION -
 - oracle.apps.cz.cio.IRuntimeNode.OPTION, B-143
- Option - oracle.apps.cz.cio.Option, B-170
- OPTION_FEATURE -
 - oracle.apps.cz.cio.IRuntimeNode.OPTION_FEATURE, B-143
- OptionFeature -
 - oracle.apps.cz.cio.OptionFeature, B-173
- OptionFeatureNode -
 - oracle.apps.cz.cio.OptionFeatureNode, B-176
- OptionNode -
 - oracle.apps.cz.cio.OptionNode, B-180
- Oracle Configurator Toolkit, 2-7
- Oracle JDBC OCI drivers, 1-8
- Oracle JDBC Thin drivers, 1-8
- Oracle Technology Network, 1-8
- oracle.apps.cz.cio, 2-1, 2-2
- oracle.apps.cz.cio - oracle.apps.cz.cio, B-1
- oracle.apps.cz.cio.IFunctionalCompanion, 2-23
- oracle.apps.cz.common -
 - oracle.apps.cz.common, C-1
- oracle.apps.cz.utilities -
 - oracle.apps.cz.utilities, D-1
- ORTHEN -
 - oracle.apps.cz.cio.Reason.ORTHEN, B-192
- Output, 1-2, 1-11, 1-15
- override()
 - usage, 2-18, 2-20
- override() -
 - oracle.apps.cz.cio.LogicalOverridableException.override(), B-157

O

onNew() -

P

PATH, 1–8
perform() -
 oracle.apps.cz.cio.Configuration.perform(), B-75
PRICE_APPS_107_110 -
 oracle.apps.cz.cio.Configuration.PRICE_APPS_107_110, B-65
PRICE_MULT_ITEMS -
 oracle.apps.cz.cio.Configuration.PRICE_MULT_ITEMS, B-65
PRICE_SINGLE_ITEM -
 oracle.apps.cz.cio.Configuration.PRICE_SINGLE_ITEM, B-65
PRICING_DISABLED -
 oracle.apps.cz.cio.Configuration.PRICING_DISABLED, B-65
PricingUnavailableException -
 oracle.apps.cz.cio.PricingUnavailableException, B-184
PricingUnavailableException(String) -
 oracle.apps.cz.cio.PricingUnavailableException.
 PricingUnavailableException(java.lang.String), B-185
PricingUnavailableException(String, Object, Log) -
 oracle.apps.cz.cio.PricingUnavailableException.
 PricingUnavailableException(java.lang.String,
 java.lang.Object,
 oracle.apps.fnd.common.Log), B-185
Property - oracle.apps.cz.cio.Property, B-186

R

ReadOnlyDecimalNode -
 oracle.apps.cz.cio.ReadOnlyDecimalNode, B-189
Reason - oracle.apps.cz.cio.Reason, B-191
Reason(int, IRuntimeNode, String) -
 oracle.apps.cz.cio.Reason.Reason(int,
 oracle.apps.cz.cio.IRuntimeNode,
 java.lang.String), B-193
Reason(Message, String, IRuntimeNode) -
 oracle.apps.cz.cio.Reason.Reason(oracle.apps.fnd.common.Message, java.lang.String,

 oracle.apps.cz.cio.IRuntimeNode), B-193
Reason(String) -
 oracle.apps.cz.cio.Reason.Reason(java.lang.String), B-193
releaseJDBCConnection() -
 oracle.apps.cz.common.CZContext.releaseJDBCConnection(), C-5
removeConfigEventListener(ICompSetEventListener) -
 oracle.apps.cz.cio.ComponentSet.removeConfigEventListener(oracle.apps.cz.cio.ICompSetEventListener), B-59
removeConfigEventListener(IConfigEventListener) -
 oracle.apps.cz.cio.Configuration.removeConfigEventListener(oracle.apps.cz.cio.IConfigEventListener), B-75
RESOURCE -
 oracle.apps.cz.cio.IRuntimeNode.RESOURCE, B-143
Resource - oracle.apps.cz.cio.Resource, B-195
RESOURCE_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.RESOURCE_FAILURE, B-230
restartConfiguration(boolean) -
 oracle.apps.cz.cio.Configuration.restartConfiguration(boolean), B-75
RESTORE_FAILURE -
 oracle.apps.cz.cio.ValidationFailure.RESTORE_FAILURE, B-231
restoreConfiguration()
 usage, 2–7
restoreConfiguration(DbConfigHeader, Context) -
 oracle.apps.cz.cio.CIO.restoreConfiguration(oracle.apps.cz.dio.config.DbConfigHeader,
 oracle.apps.fnd.common.Context), B-39
restoreConfiguration(int, Context) -
 oracle.apps.cz.cio.CIO.restoreConfiguration(int,
 int, oracle.apps.fnd.common.Context), B-40
RestoreValidationFailure -
 oracle.apps.cz.cio.RestoreValidationFailure, B-197
revNumber
 usage, 2–7
rollbackConfigTransaction()
 usage, 2–20

- rollbackConfigTransaction(ConfigTransaction) -
 oracle.apps.cz.cio.Configuration.rollbackConfig
 Transaction(oracle.apps.cz.cio.ConfigTransactio
 n), B-75
- rollbackConfigTransaction(transaction)
 usage, 2-8, 2-21
- runtime model subschema objects, 2-1
- RuntimeNode -
 oracle.apps.cz.cio.RuntimeNode, B-199

S

- save()
 usage, 2-7
- save() -
 oracle.apps.cz.cio.Configuration.save(), B-75
- saveAs(configHeaderID, revNumber)
 usage, 2-7
- saveAs(int, int) -
 oracle.apps.cz.cio.Configuration.saveAs(int,
 int), B-76
- saveNew()
 usage, 2-7
- saveNew() -
 oracle.apps.cz.cio.Configuration.saveNew(), B
 -76
- saveNewRev()
 usage, 2-7
- saveNewRev() -
 oracle.apps.cz.cio.Configuration.saveNewRev(),
 B-76
- select()
 usage, 2-15
- select() - oracle.apps.cz.cio.BomNode.select(), B-24
- select() - oracle.apps.cz.cio.IOption.select(), B-131
- select() -
 oracle.apps.cz.cio.OptionNode.select(), B-182
- select(IOption) -
 oracle.apps.cz.cio.BomNode.select(oracle.apps.c
 z.cio.IOption), B-24
- select(IOption) -
 oracle.apps.cz.cio.IOptionFeature.select(oracle.a
 pps.cz.cio.IOption), B-134
- select(IOption) -
 oracle.apps.cz.cio.OptionFeatureNode.select(or

- acle.apps.cz.cio.IOption), B-178
- SelectionNotMutexedException -
 oracle.apps.cz.cio.SelectionNotMutexedException,
 B-207
- selectOption()
 usage, 2-14
- setActiveModelPath(String) -
 oracle.apps.cz.cio.CIO.setActiveModelPath(java
 .lang.String), B-40
- setAltPricingAtpContext(Context) -
 oracle.apps.cz.cio.Configuration.setAltPricingA
 tpContext(oracle.apps.fnd.common.Context),
 B-76
- setAtpDate(Date) -
 oracle.apps.cz.cio.BomNode.setAtpDate(java.ut
 il.Date), B-24
- setConfigHeaderCheckoutUser(String) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erCheckoutUser(java.lang.String), B-77
- setConfigHeaderDateCreated(Timestamp) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erDateCreated(java.sql.Timestamp), B-77
- setConfigHeaderDescription(String) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erDescription(java.lang.String), B-77
- setConfigHeaderEffectiveFrom(Timestamp) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erEffectiveFrom(java.sql.Timestamp), B-77
- setConfigHeaderEffectiveTo(Timestamp) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erEffectiveTo(java.sql.Timestamp), B-77
- setConfigHeaderName(String) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erName(java.lang.String), B-77
- setConfigHeaderNote(String) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erNote(java.lang.String), B-77
- setConfigHeaderOpportunityHeaderId(int) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erOpportunityHeaderId(int), B-77
- setConfigHeaderUiDefinitionId(int) -
 oracle.apps.cz.cio.Configuration.setConfigHead
 erUiDefinitionId(int), B-77
- setCount(int) -
 oracle.apps.cz.cio.ICount.setCount(int), B-108

- setCount(int) -
 - oracle.apps.cz.cio.StateCountNode.setCount(int), B-212
- setCurrLang(String) -
 - oracle.apps.cz.common.CZContext.setCurrLang(java.lang.String), C-5
- setDecimalValue()
 - usage, 2-13
- setDecimalValue(double) -
 - oracle.apps.cz.cio.DecimalNode.setDecimalValue(double), B-87
- setDecimalValue(double) -
 - oracle.apps.cz.cio.IDecimal.setDecimalValue(double), B-110
- setDiscountedPrice(double) -
 - oracle.apps.cz.cio.StateCountNode.setDiscountedPrice(double), B-213
- setInitParameters(NameValuePairSet) -
 - oracle.apps.cz.cio.Configuration.setInitParameters(oracle.apps.cz.utilities.NameValuePairSet), B-78
- setIntValue()
 - usage, 2-13
- setIntValue(int) -
 - oracle.apps.cz.cio.CountFeature.setIntValue(int), B-82
- setIntValue(int) -
 - oracle.apps.cz.cio.IInteger.setIntValue(int), B-119
- setIntValue(int) -
 - oracle.apps.cz.cio.IntegerNode.setIntValue(int), B-128
- setListPrice(double) -
 - oracle.apps.cz.cio.StateCountNode.setListPrice(double), B-213
- setName(String) -
 - oracle.apps.cz.cio.Component.setName(java.lang.String), B-54
- setState()
 - usage, 2-12
- setState(int state)
 - usage, 2-12
- setState(int) -
 - oracle.apps.cz.cio.BomNode.setState(int), B-2

4

- setState(int) -
 - oracle.apps.cz.cio.IState.setState(int), B-150
- setState(int) -
 - oracle.apps.cz.cio.OptionNode.setState(int), B-182
- setState(int) -
 - oracle.apps.cz.cio.StateNode.setState(int), B-216
- setState(state)
 - usage, 2-18
- setTextValue(String) -
 - oracle.apps.cz.cio.IText.setTextValue(java.lang.String), B-151
- setTextValue(String) -
 - oracle.apps.cz.cio.TextNode.setTextValue(java.lang.String), B-224
- setValue(Object) -
 - oracle.apps.cz.utilities.NameValuePair.setValue(java.lang.Object), D-3
- startDeltaList() -
 - oracle.apps.cz.cio.Configuration.startDeltaList(), B-78
- StateCountNode -
 - oracle.apps.cz.cio.StateCountNode, B-209
- StateNode - oracle.apps.cz.cio.StateNode, B-214
- statesMatch(int, int) -
 - oracle.apps.cz.cio.StateNode.statesMatch(int, int), B-217
- STATUS_DELETED -
 - oracle.apps.cz.cio.StatusInfo.STATUS_DELETED, B-219
- STATUS_EXISTING -
 - oracle.apps.cz.cio.StatusInfo.STATUS_EXISTING, B-219
- STATUS_NEW -
 - oracle.apps.cz.cio.StatusInfo.STATUS_NEW, B-219
- StatusInfo - oracle.apps.cz.cio.StatusInfo, B-218
- statusToString(int) -
 - oracle.apps.cz.cio.StatusInfo.statusToString(int), B-220
- subschema objects
 - configuration, 2-1
 - runtime model, 2-1

T

terminate()
 usage, 2–28
terminate() -
 oracle.apps.cz.cio.FunctionalCompanion.terminate(), B–99
terminate() -
 oracle.apps.cz.cio.IFunctionalCompanion.terminate(), B–117
TEXT - oracle.apps.cz.cio.Reason.TEXT, B–192
TEXT_FEATURE -
 oracle.apps.cz.cio.IRuntimeNode.TEXT_FEATURE, B–143
TextFeature - oracle.apps.cz.cio.TextFeature, B–221
TextNode - oracle.apps.cz.cio.TextNode, B–223
TOGGLE
 usage, 2–11
TOGGLE -
 oracle.apps.cz.cio.IState.TOGGLE, B–149
toString()
 usage, 2–19
toString() -
 oracle.apps.cz.cio.DecimalNode.toString(), B–87
toString() -
 oracle.apps.cz.cio.FunctionalCompanion.toString(), B–99
toString() -
 oracle.apps.cz.cio.ReadOnlyDecimalNode.toString(), B–190
toString() -
 oracle.apps.cz.cio.Reason.toString(), B–194
toString() -
 oracle.apps.cz.cio.RuntimeNode.toString(), B–205
toString() -
 oracle.apps.cz.cio.StateCountNode.toString(), B–213
toString() -
 oracle.apps.cz.cio.StatusInfo.toString(), B–220
toString() -
 oracle.apps.cz.cio.ValidationFailure.toString(), B–231
toString(boolean) -

 oracle.apps.cz.cio.IRuntimeNode.toString(boolean), B–147
toString(boolean) -
 oracle.apps.cz.cio.RuntimeNode.toString(boolean), B–206
toString(boolean) -
 oracle.apps.cz.cio.StatusInfo.toString(boolean), B–220
TOTAL -
 oracle.apps.cz.cio.IRuntimeNode.TOTAL, B–143
Total - oracle.apps.cz.cio.Total, B–225
trace(String) -
 oracle.apps.cz.cio.CIO.trace(java.lang.String), B–41
tracing() - oracle.apps.cz.cio.CIO.tracing(), B–41
TransactionException -
 oracle.apps.cz.cio.TransactionException, B–227
transactions
 logic, 2–8
translate() -
 oracle.apps.cz.cio.Reason.translate(), B–194
translate(String) -
 oracle.apps.cz.cio.Reason.translate(java.lang.String), B–194
TRUE
 usage, 2–11
TRUE - oracle.apps.cz.cio.IState.TRUE, B–149
TRUEATBIRTH -
 oracle.apps.cz.cio.Reason.TRUEATBIRTH, B–193
typeToString(int) -
 oracle.apps.cz.cio.RuntimeNode.typeToString(int), B–206

U

UFALSE
 usage, 2–11
UFALSE - oracle.apps.cz.cio.IState.UFALSE, B–149
undo()
 usage, 2–14
undo() -
 oracle.apps.cz.cio.Configuration.undo(), B–78

UNKNOWN

usage, 2–12

UNKNOWN -

oracle.apps.cz.cio.IState.UNKNOWN, B-150

unset() -

oracle.apps.cz.cio.DecimalNode.unset(), B-88

unset() - oracle.apps.cz.cio.ICount.unset(), B-109

unset() - oracle.apps.cz.cio.IDecimal.unset(), B-111

unset() - oracle.apps.cz.cio.IInteger.unset(), B-119

unset() -

oracle.apps.cz.cio.IntegerNode.unset(), B-129

unset() - oracle.apps.cz.cio.IState.unset(), B-150

unset() - oracle.apps.cz.cio.IText.unset(), B-152

unset() -

oracle.apps.cz.cio.StateNode.unset(), B-217

unset() -

oracle.apps.cz.cio.TextNode.unset(), B-224

UTRUE

usage, 2–12

UTRUE - oracle.apps.cz.cio.IState.UTRUE, B-150

V

validate()

usage, 2–10, 2–22, 2–26

validate() -

oracle.apps.cz.cio.FunctionalCompanion.validate(), B-99

validate() -

oracle.apps.cz.cio.IFunctionalCompanion.validate(), B-117

Validation, 1–2, 1–11, 1–15, 2–10

ValidationFailure -

oracle.apps.cz.cio.ValidationFailure, B-229

W

web deployment, 2–7, 2–28, 4–9