Oracle Utilities Quotations Management

Installation and Configuration Guide

Release 1.6.1.23 for Windows

E18215-24

December 2018

(Revised April 2020)

ORACLE

Oracle Utilities Quotations Management/Quotations Management Installation and Configuration Guide, Release 1.6.1.23 for Windows

E18215-24

Primary Author: Lou Prosperi

Contributor: Steve Pratt

NOTIFICATION OF THIRD-PARTY LICENSES

Oracle Utilities software contains third party, open source components as identified below. Third- party license terms and other third-party required notices are provided below.

License: Apache 1.1

Module: xercesImpl.jar, xalan.jar

Use of xercesImpl and xalan within the product is governed by the following (Apache 1.1):

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/) ." Alternately, this acknowledgment may appear in the software itself, if and

wherever such third-party acknowledgments normally appear. (4) Neither the component name nor Apache Software Foundation may be used to endorse or promote products derived from the software without specific prior written permission. (5) Products derived from the software may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission.

THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License: Paul Johnston

Modules: md5.js

Copyright (C) Paul Johnston 1999 - 2002

Use of these modules within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution. (3) Neither the component name nor the names of the copyright holders and contributors may be used to endorse or promote products derived from the software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License: Tom Wu

Module: jsbn library

Copyright © 2003-2005 Tom Wu. All rights reserved

Use of this module within the product is governed by the following:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below. (2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer below in the documentation and/or other materials provided with the distribution.

THE SOFTWARE IS PROVIDED “AS-IS” AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL TOM WU BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Contents

Contents

What's New

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 6

Chapter 7

Chapter 8

Appendix A

Index

What's New

New Features in the Oracle Utilities Quotations Management Installation and Configuration Guide

This chapter outlines the new features of the 1.6.0.0 release of Oracle Utilities Quotations Management that are documented in this guide.

New Features for Release 1.6.0.0

Feature	Description	For more information, refer to...
Term-Based Rules Language Functions	This release includes new Rules Language functions to retrieve and save terms and term details used in contract calculations to and from the Oracle Utilities Data Repository.	**Oracle Utilities Rules Language - Term Functions** on page 7-7

Chapter 1

Overview

This chapter provides an overview of the installation and configuration of Oracle Utilities Quotations Management, including:

- **Installation and Configuration Overview**
- **What is this book?**

Installation and Configuration Overview

Installing and configuring Oracle Utilities Quotations Management involves the following steps:

- Setup the Oracle Utilities Quotations Management Network Environment as described in **Chapter Two: Setting Up the Network Environment**.
- Install the Oracle Utilities Quotations Management Database as described in **Chapter 2: Oracle Utilities Quotations Management Network Environment and Database Installation** and **Chapter Three: Oracle Utilities Data Repository Schema Creation** of the *Oracle Utilities Energy Information Platform Configuration Guide*.
- Install workstation and application server applications used by Oracle Utilities Quotations Management on client machines as described in **Chapter 3**: **Installing the Oracle Utilities Quotations Management Application Software**.
- Install Oracle Utilities Quotations Management Web components on web server as described in **Chapter 3**: **Installing the Oracle Utilities Quotations Management Application Software**.
- Install Oracle Utilities Quotations Management database records and rate forms as described in **Chapter 3**: **Installing the Oracle Utilities Quotations Management Application Software**
- Set up and configure database records as described in **Chapter 5**: **Setting Up Contract Database Tables**.
- Set up and configure Oracle Utilities Quotations Management products and terms as described in **Chapter 6**: **Setting Up Terms, Products, and Contract Types**.
- Set up and configure contract types as described in **Chapter 6**: **Setting Up Terms, Products, and Contract Types**.
- Add customized business logic to Oracle Utilities Quotations Management as described in **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management**.
- Set up and configure security for use with Oracle Utilities Quotations Management as described in **Chapter 8**: **Configuring Oracle Utilities Quotations Management Security**.

What is this book?

The *Oracle Utilities Quotations Management Installation and Configuration Guide* describes how to install and configure Oracle Utilities Quotations Management, including the following:

- **Chapter 1**: **Overview** (this chapter) provides an overview of the Oracle Utilities Quotations Management installation and configuration process
- **Chapter 2**: **Oracle Utilities Quotations Management Network Environment and Database Installation** describes how to install the Oracle Utilities Quotations Management Database.
- **Chapter 3**: **Installing the Oracle Utilities Quotations Management Application Software** describes how to install Oracle Utilities Quotations Management application software on an application server, on a web server, and on client machines.
- **Chapter 4**: **Oracle Utilities Quotations Management Rules Language Schedules, Riders, and Lists** describes the Oracle Utilities Rules Language rate schedules and riders used by the Oracle Utilities Quotations Management application.
- **Chapter 5**: **Setting Up Contract Database Tables** describes how to set up and configure contract tables used by the Oracle Utilities Quotations Management application.
- **Chapter 6**: **Setting Up Terms, Products, and Contract Types** describes how to set up and configure products and terms used by the Oracle Utilities Quotations Management application.
- **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** describes how to add customized business logic to Oracle Utilities Quotations Management.
- **Chapter 8**: **Configuring Oracle Utilities Quotations Management Security** describes how to configure security for use with Oracle Utilities Quotations Management.
- **Appendix A**: **Oracle Utilities Data Repository Quotations Management Database Schema** provides a database schema diagram of the database tables used with Oracle Utilities Quotations Management.

Chapter 2

Oracle Utilities Quotations Management Network Environment and Database Installation

This chapter describes the Oracle Utilities Quotations Management network environment and how to install the Oracle Utilities Date Repository used by Oracle Utilities Quotations Management, including:

- **The Oracle Utilities Quotations Management Installation Package**
- **Oracle Utilities Quotations Management Network Environment**
- **Oracle Utilities Quotations Management Database Installation**

The Oracle Utilities Quotations Management Installation Package

There is a separate installation package for the Oracle Utilities Energy Information Platform and for each related product. The table below provides the installation package file names for the Oracle Utilities Energy Information Platform and Oracle Utilities Quotations Management.

Product	Installation File Name
Oracle Utilities Energy Information Platform	1.6.1.xx.0.EIP.zip
Oracle Utilities Quotations Management	1.6.1.xx.0.QM.zip

The Oracle Utilities Quotations Management installation package contains the following folders:

- **Install**: Contains the installation program for Oracle Utilities Quotations Management, including:
  - Oracle Utilities QM 1.6.1.xx.0.msi
- **DBScripts**: Contains the following database scripts for Oracle Utilities Quotations Management:
  - Install: scripts used to create a new Quotations Management schema
  - Upgrade: scripts used to update the Quotations Management schema

  See **Oracle Utilities Quotations Management Database Installation** on page 2-9 for more information about installing and upgrading the Oracle Utilities Quotations Management database schema.
- **Documentation**: Contains documentation for Oracle Utilities Quotations Management, including:
  - Oracle Utilities Quotations Management Installation and Configuration Guide
  - Oracle Utilities Quotations Management User's Guide

    Note: Oracle Utilities Quotations Management documentation can also be downloaded separately.

Oracle Utilities Quotations Management Network Environment

This section contains options for hardware and software specifications for implementation of the Oracle Utilities Energy Information Platform and related products, including:

- Oracle Utilities Billing Component
- Oracle Utilities Load Analysis
- Oracle Utilities Load Profiling and Settlement
- Oracle Utilities Meter Data Management
- Oracle Utilities Quotations Management
- Oracle Utilities Rate Management
- Oracle Utilities Transaction Management

The Energy Information Platform application architecture comprises all of the components typically found in any n-tier enterprise architecture, including client machines, servers, and the supporting network infrastructure.

The Oracle Utilities Energy Information Platform uses five primary types of systems components. Suggested hardware specifications and required software for each component are listed below.

Before installing Oracle Utilities software you should consult with Oracle Utilities Consulting. They will make suggestions for your hardware and network architecture based on your business requirements. You should also consult with Oracle Utilities Consulting if your business requirements change, if your database grows significantly, or if you plan on migrating to a different version of Oracle Utilities software.

> **Note:** Supported software versions are subject to change. Please refer to the Release Notes shipped with the software for the latest supported versions.

Client Workstation - C/S

These are workstations used when running the client/server (C/S) versions of Oracle Utilities applications, such as Data Manager.

Software for Client Workstation - C/S:

- Supported Operating Systems
  - Windows 8.1
  - Windows 10
- Supported Web Browsers
  - Microsoft Internet Explorer 11 with latest security updates
  - Microsoft Edge (Windows 10 only)
- Relational Database Management System (RDBMS) client with applicable ODBC driver and Data Provider for .NET

  Note: When using 64-bit operating systems, the 32-bit version of the database client is required.
- Oracle Utilities Energy Information Platform software
- Microsoft .NET Framework 4.5
- Microsoft Visual C++ Redistributable Packages for Visual Studio 2013 (32-bit)
- Microsoft Data Access Component (MDAC) 2.8 or higher
- Microsoft XML Core Services (MSXML) 6.0 or higher

Client Workstation - Web

These are workstations used when running the web-enabled versions of Oracle Utilities applications only.

Software for Client Workstation - Web:

- Supported Operating Systems
  - Windows 8.1
  - Windows 10
- Supported Web Browsers
  - Microsoft Internet Explorer 11 with latest security updates
  - Microsoft Edge (Windows 10 only)

Note: For users that require access to both client/server and web browser-based applications, please refer to the requirements listed for "Client Workstations - Thick Client."

Database Server

The database server houses the Oracle Utilities Data Repository. The hardware platform for the database server may be Windows/Intel or UNIX/RISC.

Software for Database Server:

- Supported Operating Systems
  - Microsoft Windows Server 2012
  - Microsoft Windows Server 2016
  - LINUX (UNIX/Intel)
  - Sun Microsystems Solaris
  - HP-UX
  - IBM AIX (UNIX/RISC) software

  Note: Refer to the *Oracle Utilities Energy Information Platform Quick Install Guide* for additional details about supported operating systems.
- RDBMS Server: Oracle

Database Platforms:

- Oracle 11g, Release 2 (11.2.0.4)
- Oracle 12c, Release 1 (12.1.0.2 or 12.2.0.1)
- Oracle 19c, Release 1 (19.3.0.0.0)

Note: Supported database versions and drivers are subject to change. Please refer to the Oracle Release Notes shipped with the software for the latest compatibility matrix.

Application/Batch Processing Server

Application servers are used when running Windows Services, such as those employed by the Reporting Framework and Adapter components. The application server(s) can also perform batch processes such as meter data upload and validation as well as batch billing and settlements.

Software for the Application/Batch Server:

- Supported Operating Systems
  - Windows 2012 Server R2
  - Windows 2016 Server
- Supported Web Browsers
  - Microsoft Internet Explorer 11 with latest security updates
  - Microsoft Edge with latest security updates (Windows 2016 Server only)
- RDBMS client with applicable ODBC driver and Data Provider for .NET

  Note: When using 64-bit operating systems, the 32-bit version of the database client is required.
- Oracle Utilities Energy Information Platform software
- Microsoft .NET Framework 4.5
- Microsoft Visual C++ Redistributable Packages for Visual Studio 2013 (32-bit)
- Microsoft Data Access Component (MDAC) 2.8 or higher
- Microsoft XML Core Services (MSXML) 6.0 or higher
- Java Development Kit (JDK) 1.8 or later (for application servers that will be running the Adapter)

Web Server(s)

Web servers run Microsoft Internet Information Service (IIS). These servers may be combined with the application server(s) for combination Web and Application servers.

Software for Web Server(s):

- Supported Operating Systems
  - Windows 2012 R2 Enterprise Server with Microsoft Internet Information Service (IIS) 8.5
  - Windows 2016 Enterprise Server with Microsoft Internet Information Service (IIS) 10.0
- Supported Web Browsers
  - Microsoft Internet Explorer 11 with latest security updates
  - Microsoft Edge with latest security updates (Windows 2016 Server only)
- RDBMS client with applicable ODBC driver and Data Provider for .NET

  Note: When using 64-bit operating systems, the 32-bit version of the database client is required.
- Oracle Utilities Energy Information Platform software
- Microsoft .NET Framework 4.5
- Microsoft Visual C++ Redistributable Packages for Visual Studio 2013 (32-bit)
- Microsoft Data Access Component (MDAC) 2.8 or higher
- Microsoft XML Core Services (MSXML) 6.0 or higher

Oracle Utilities Quotations Management Database Installation

This section describes how to install and verify the Oracle Utilities Quotations Management database tables and data in the Oracle Utilities Data Repository, including:

- **Installation Requirements**
- **Installing the Database**
- **Upgrading the Quotations Management Database Schema**
- **Updating the Reporting Framework for Oracle Business Intelligence Publisher**
- **Verifying the Database**

> **Note**: This section assumes that you have created the Oracle Utilities Data Repository schema as described in **Chapter 3**: **Oracle Utilities Data Repository Schema Creation** in the *Oracle Utilities Energy Information Platform Installation Guide*.

Installation Requirements

The following are required in order to install the Oracle Utilities Quotations Management tables and data into the Oracle Utilities Data Repository:

- The Oracle Utilities Data Repository (v1.6.1.0.0) schema must have been installed on the database instance on which you plan to run the Oracle Utilities Quotations Management application.
- The **addCE.cmd** file. This database script adds the contract database tables and data used by Oracle Utilities Quotations Management to the Oracle Utilities Data Repository.
- The **updateCE.cmd** file. This database script updates the contract management tables and data used by Oracle Utilities Quotations Management in the Oracle Utilities Data Repository. See **Upgrading the Quotations Management Database Schema** on page 2-10 for more information.
- The **BIPCE.cmd** file. This database script changes the pre-defined contract management reports from Crystal Reports to Oracle Business Intelligence Publisher. See **Updating the Reporting Framework for Oracle Business Intelligence Publisher** on page 2-12 for more information.
- The **addUE.cmd** file. This database script adds the usage database tables and data used by Oracle Utilities Quotations Management to the Oracle Utilities Data Repository.

Installing the Database

Installing the Oracle Utilities Quotations Management database involves installing the Oracle Utilities Billing Component - Contract Management database schema and the Usage database schema into the Oracle Utilities Date Repository.

Oracle Utilities Billing Component - Contract Management Schema

Open a command prompt and run the addCE.CMD script. This script uses the following syntax:

```
addCE [-d <database>] [-own <owner name>] -opw <owner password>
```

Parameter	Description
<database>	The name given to the instance as specified in the TNSNAMES.ORA file. This parameter is optional and if not specified, the script will connect to the default Oracle database.

Parameter	Description
<owner name>	The name of the user which will own Oracle Utilities database objects. This parameter is optional. If not specified, the default Oracle Utilities user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Like the base schema database tables and indexes, the Oracle Utilities Billing Component - Contract Management objects are created in the default tablespace of the PWRLINE user with default sizing parameters. If these defaults are required to be changed then the scripts may be edited.

Usage Schema

Open a command prompt and run the addUE.CMD script. This script uses the following syntax:

```
addUE [-d <database>] [-own <owner name>] -opw <owner password>
```

Parameter	Description
<database>	The name given to the instance as specified in the TNSNAMES.ORA file. This parameter is optional and if not specified, the script will connect to the default Oracle database.
<owner name>	The name of the user which will own Oracle Utilities database objects. This parameter is optional. If not specified, the default Oracle Utilities user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Like the base schema database tables and indexes, the Usage objects are created in the default tablespace of the PWRLINE user with default sizing parameters. If these defaults are required to be changed then the scripts may be edited.

Upgrading the Quotations Management Database Schema

If you are upgrading Oracle Utilities Quotations Management from a previous version, you must upgrade Quotations Management database schema. The following database upgrade scripts are included in the Quotations Management installation package.

- The **updateCE.cmd** file. This database script updates the contract management tables and data used by Oracle Utilities Quotations Management in the Oracle Utilities Data Repository. This script can only be used when upgrading the following schema version.

v1.6 Schema Version	v1.6.1 Schema Version
v1.6.0.0.0 (Contract Management)	v1.6.1.0.0

  If you are upgrading from a different schema version, contact Oracle Global Customer Support.

- The **updateUE.cmd** file. This database script updates the usage schema tables and data used by Oracle Utilities Quotations Management in the Oracle Utilities Data Repository. This script can only be used when upgrading the following schema version.

v1.6 Schema Version	v1.6.1 Schema Version
v1.6.0.0.0 (Usage)	v1.6.1.2.0

If you are upgrading from a different schema version, contact Oracle Global Customer Support.

Upgrading Oracle Utilities Quotations Management Schemas

Open a command prompt and run the updateCE.CMD script. This script uses the following syntax:

```
updateCE [-d <database>] [-own <owner name>] -opw <owner password>
```

Parameter	Description
<database>	The name given to the instance as specified in the TNSNAMES.ORA file. This parameter is optional and if not specified, the script will connect to the default Oracle database.
<owner name>	The name of the user which will own Oracle Utilities database objects. This parameter is optional. If not specified, the default Oracle Utilities user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Open a command prompt and run the updateUE.CMD script. This script uses the following syntax:

```
updateUE [-d <database>] [-own <owner name>] -opw <owner password>
```

Parameter	Description
<database>	The name given to the instance as specified in the TNSNAMES.ORA file. This parameter is optional and if not specified, the script will connect to the default Oracle database.
<owner name>	The name of the user which will own Oracle Utilities database objects. This parameter is optional. If not specified, the default Oracle Utilities user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Updating the Reporting Framework for Oracle Business Intelligence Publisher

If you are using Oracle Business Intelligence Publisher for reporting, you must update the pre-defined reports in Reporting Framework database tables. The following database scripts are included in the Quotations Management installation package for this purpose.

- The **BIPCE.cmd** file. This script changes the pre-defined contract management reports from Crystal Reports to Oracle Business Intelligence Publisher.

Oracle Utilities Quotations Management Reports

Open a command prompt and run the BIPCE.CMD script. This script uses the following syntax:

```
BIPCE [-d <database>] [-own <owner name>] -opw <owner password>
```

Parameter	Description
<database>	The name given to the instance as specified in the TNSNAMES.ORA file. This parameter is optional and if not specified, the script will connect to the default Oracle database.
<owner name>	The name of the user which will own Oracle Utilities database objects. This parameter is optional. If not specified, the default Oracle Utilities user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Verifying the Database

Verifying the Oracle Utilities Quotations Management database involves verifying the Oracle Utilities Billing Component - Contract Management database schema, and the Usage database schema in the Oracle Utilities Date Repository.

Verification - Oracle Utilities Billing Component - Contract Management

To verify that the Oracle Utilities Billing Component - Contract Management tables are in place, use the following procedure:

1. Log into the database using the PWRLINE user (Password =password).
2. Verify that the following tables exist in the database:
    - CM Channel Cut Data
    - CM Channel Cut Header
    - CM Channel Cut Validation Messages
    - Contract
    - Contract Category
    - Contract Document
    - Contract External Status
    - Contract History
    - Contract Item
    - Contract Item Details
    - Contract Item List
    - Contract Item List Relationship
    - Contract Item List Term
    - Contract Item Product Term
    - Contract Item Term
    - Contract Relationship
    - Contract Status
    - Contract Term
    - Contract Type
    - Contract Type Doc Rel
    - Contract Type Term
    - Contractee
    - Contractee Directory
    - Contractee Status
    - Contract Type
    - Document
    - Document Category
    - Document Status
    - Document Type
    - Product

- Product Term
- Term
- Term Category
- Term Status
- Term Type

Verification - Usage

To verify that the Usage tables are in place, use the following procedure:

1. Log into the database using the PWRLINE user (Password =password).
2. Verify that the following tables exist in the database:
   - Forecast (used with Oracle Utilities Quotations Management)
   - Forecast Status (used with Oracle Utilities Quotations Management)
   - Forecast Value (used with Oracle Utilities Quotations Management)
3. In addition, verify that the Channel History table contains the following new columns:
   - Usage Data
   - Status
   - Validation Method
   - Validation Message
   - Estimation Required?
   - Forecast

Appendix A: **Oracle Utilities Data Repository Quotations Management Database Schema** includes a diagram of the Oracle Utilities Data Repository Quotations Management database schema (v1.6.1.0.0) that provides details regarding the table and columns in the Oracle Utilities Quotations Management schema, as well as the relationships between these tables in the Oracle Utilities Data Repository.

Verification - Contract Management Reports

To verify that the contract management reports have been updated, use the following procedure:

1. Log into the database using the PWRLINE user (Password =password).
2. Verify that the following records exist in the Report Instance table:

Report Name	Description	Report Type
ViewCalculations	View Calculations	BIPublisher
ViewContract	View Contract	BIPublisher

Chapter 3

Installing the Oracle Utilities Quotations Management Application Software

This chapter describes how you install the Oracle Utilities Quotations Management application software, including:

- **Installing the Oracle Utilities Quotations Management Software**
- **Setting Up Configuration Files on the Web Server**
- **Setting Up Configuration Files on Application Server and Workstations**
- **Installing Oracle Utilities Quotations Management Database Records and Rate Forms**

Installing the Oracle Utilities Quotations Management Software

This section describes how to install the Oracle Utilities Quotations Management software including:

- **Installing in Conjunction with the Energy Information Platform**
- **Installing After Installation of the Energy Information Platform**

Installing in Conjunction with the Energy Information Platform

To install the Oracle Utilities Quotations Management software in conjunction with the Oracle Utilities Energy Information Platform, use the following procedure:

1. Navigate to the Install folder created by the Energy Information Platform installation package (1.6.1.xx.0.EIP.zip). This folder contains the following files:
   - Oracle Utilities EIP 1.6.1.xx.0.msi
   - setup.exe
2. Navigate to the Install folder created by the Oracle Utilities Quotations Management installation package (1.6.1.xx.0.QM.zip). This file contains the following files:
   - Oracle Utilities QM 1.6.1.xx.0.msi
3. Move the "Oracle Utilities QM 1.6.1.xx.0.msi" file into the same directory as the Energy Information Platform files.

   The directory should now contain following:
   - Oracle Utilities EIP 1.6.1.xx.0.msi
   - setup.exe
   - Oracle Utilities QM 1.6.1.xx.0.msi
4. Double-click the setup.exe file.

   A dialog opens asking you to confirm the products you wish to install. Click **Yes** to continue with the installation. Click **No** to cancel the installation.
5. Proceed with the installation (starting at Step 3) as outlined in **Chapter 4**: **Installing the Oracle Utilities Application Software** in the *Oracle Utilities Energy Information Platform Installation Guide*.

Installing After Installation of the Energy Information Platform

To install the Oracle Utilities Quotations Management software after the Oracle Utilities Energy Information Platform software has been installed, use the following procedure:

1. Navigate to the Install folder created by the Oracle Utilities Quotations Management installation package (1.6.1.xx.0.QM.zip). This file contains the following files:
   - Oracle Utilities QM 1.6.1.xx.0.msi
2. Double-click the Oracle Utilities QM 1.6.1.xx.0.msi file.

   A dialog opens asking you to confirm the products you wish to install. Click **Yes** to continue with the installation. Click **No** to cancel the installation.
3. Proceed with the installation (starting at Step 3) as outlined in **Chapter 4**: **Installing the Oracle Utilities Application Software** in the *Oracle Utilities Energy Information Platform Installation Guide*.

Setting Up Configuration Files on the Web Server

Oracle Utilities web applications use the following configuration files.

LODESTAR.CFG

The LODESTAR.CFG file is a text file used to customize the working environment of the Oracle Utilities application software. See **LODESTAR.CFG** on page 2-2 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

> **Note:** This file **MUST** be named LODESTAR.CFG on the web server.

LSSECURE.CFG.XML

The LSSECURE.CFG.XML file specifies the data source used by the Security functionality and is required in order to run Oracle Utilities web-enabled applications. The LSSECURE.CFG.XML must be installed in the **C:\LODESTAR\CFG** directory. See **LSSECURE.CFG.XML** on page 2-42 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

> **Note:** If an application server and web server are installed on different machines, the LSSECURE.CFG.XML file is installed in the C:\LODESTAR\CFG directory with the web server components.

LSREPORTMONITOR.CFG.XML (optional)

The LSREPORTMONITOR.CFG.XML file specifies where report data is stored and how reports are processed through the web-enabled Oracle Utilities Energy Information Platform. The LSREPORTMONITOR.CFG.XML must be installed in the **C:\LODESTAR\CFG** directory. See **LSREPORTMONITOR.CFG.XML** on page 2-36 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

> **Note:** If an application server and web server are installed on different machines, the LSREPORTMONITOR.CFG.XML file is installed in the C:\LODESTAR\CFG directory on both server.

LSLOGGER.CFG.XML (optional)

The LSLOGGER.CFG.XML file defines how log files are generated by Oracle Utilities applications. Each Oracle Utilities component can be configured to write messages to a specified log file or an Event Log. The LSLOGGER.CFG.XML must be installed in the **C:\LODESTAR\CFG** directory. See **LSLOGGER.CFG.XML** on page 2-27 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

> **Note:** If an application server and web server are installed on different machines, the LSLOGGER.CFG.XML file is installed in the C:\LODESTAR\CFG directory on both machines.

LSSCHDLR.CFG.XML (optional)

The LSSCHDLR.CFG.XML file defines the data source(s) monitored by the Oracle Utilities Schedule Service (LSSCHDLR.exe). This service monitors the Scheduled Message (SCHEDULEDMESSAGE) table in each data source specified in this file. This file is required if the Oracle Utilities Schedule Service (LSSCHDLR.exe) is used. The LSSCHDLR.CFG.XML must

be installed in the **C:\LODESTAR\CFG** directory. See **LSSCHDLR.CFG.XML** on page 2-41 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

> **Note:** If an application server and web server are installed on different machines, the LSSCHDLR.CFG.XML file is installed in the C:\LODESTAR\CFG directory on both machines.

LSRELAY.CFG.XML (optional)

The LSRELAY.CFG.XML file identifies the machine running an SMTP service used to send email messages from the Oracle Utilities Energy Information Platform. This file is required only if the email functions are used. The LSRELAY.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory. See **LSRELAY.CFG.XML** on page 2-34 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

LOCALES.CFG.XML

The LOCALES.CFG.XML file specifies the regional locales available to Oracle Utilities web-enabled applications. The LOCALES.CFG.XML file must be installed in the **C:\LODESTAR\Web\CCS** directory. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server. See **LOCALES.CFG.XML** on page 2-22 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about this file.

> **Note:** If an application server and web server are installed on different machines, the LOCALES.CFG.XML file is installed in the C:\LODESTAR\Web\ccs directory with the web server components.

Setting Up Configuration Files on Application Server and Workstations

You also need to set up the configuration files needed by application server and workstation applications such as Data Manager and batch executables.

LODESTAR.CFG

The LODESTAR.CFG (also called POWRLINE.CFG in older installations) file is a text file used to customize the working environment of the Oracle Utilities application software. See **LODESTAR.CFG** on page 2-2 of the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file.

LSRELAY.CFG.XML (optional)

The LSRELAY.CFG.XML file identifies the machine running an SMTP service used to send email messages from the Oracle Utilities Energy Information Platform. This file is required only if the email functions are used. The LSRELAY.CFG.XML file must be installed in the **C:\LODESTAR\CFG** directory on any application server or workstation used to process Rules Language configuration that uses the EMAILCLIENT function. See **LSRELAY.CFG.XML** on page 2-34 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up this file. A sample of this file can be found in the **C:\LODESTAR\CFG\Examples\CFG** directory on the web server.

Installing Oracle Utilities Quotations Management Database Records and Rate Forms

The web-enabled Oracle Utilities Quotations Management application requires a specific set of database records and Rules Language rate forms be in place in the Oracle Utilities Data Repository. These records are installed in the **C:\LODESTAR\Web\cm\rateschedules** (noted as CM below) and **C:\LODESTAR\Web\pe\rateschedules** (noted as PE below) directories on the web server where the Oracle Utilities Quotations Management web application was installed, and include the following:

Rate Schedules

- LSCM_CALCULATE_PRICE.prg (CM)
- LSCM_CALCULATE_TERMS.prg (CM)
- LSCM_CREATE_CONTRACT.prg (CM)
- LSCM_CONTRACT_APPROVALS.prg (CM)
- LSCM_EXECUTE_CONTRACT.prg (CM)
- LSCM_REVISE_CONTRACT.prg (CM)
- LSCM_SHARE_ITEM_TERMS.prg (CM)
- LSCM_SUBMIT_CONTRACT.prg (CM)
- LSCM_TERMINATE_CONTRACT.prg (CM)
- LSPE_EST_USAGE.prg (PE)
- LSPE_VAL_USAGE.prg (PE)
- LSPE_VAL_USAGE_LOOSE.prg (PE)

Riders

- LSCM_ADD_HISTORY_RDR.prg (CM)
- LSCM_CALCULATE_PRICE_RDR.prg (CM)
- LSCM_CALCULATE_TERMS_CUST_RDR.prg (CM)
- LSCM_CALCULATE_TERMS_RDR.prg (CM)
- LSCM_EXECUTE_CONTRACT_RDR.prg (CM)
- LSCM_LOAD_ATTRIBUTES_RDR.prg (CM)
- LSCM_LOAD_ITEM_ATTRIBUTES_RDR.prg (CM)
- LSCM_REVISE_ATTRIBUTES_RDR.prg (CM)
- LSCM_REVISE_ITEM_ATTRIBUTES_RDR.prg (CM)
- LSCM_SHARE_ITEM_TERMS_RDR.prg (CM)
- LSCM_SUBMIT_CONTRACT_RDR.prg (CM)
- LSCM_TERMINATE_CONTRACT_RDR.prg (CM)
- LSPE_EST_USAGE_RDR.prg (PE)
- LSPE_NORM_USAGE_RDR.prg (PE)
- LSPE_VAL_USAGE_LOOSE_RDR.prg (PE)
- LSPE_VAL_USAGE_RDR.prg (PE)

Lists

- LSCM_GET_ACCOUNTS.qry (CM)
- LSCM_GET_CONTRACT.qry (CM)
- LSCM_GET_CONTRACT_CHILD.qry (CM)
- LSCM_GET_CONTRACT_DOC.qry (CM)
- LSCM_GET_CONTRACT_ITEM.qry (CM)
- LSCM_GET_CONTRACT_PARENT.qry (CM)
- LSCM_GET_CONTRACT_TERM.qry (CM)
- LSCM_GET_CONTRACTTYPE_DOC.qry (CM)
- LSCM_GET_CONTRACT_TYPE_TERM.qry (CM)
- LSCM_GET_CUSTOMER.qry (CM)
- LSCM_GET_FORECASTS.qry (CM)
- LSCM_GET_PRODUCTS.qry (CM)
- LSCM_GET_WQTYPE.qry (CM)
- LSCM_LAST_CONTRACT_REVISION.qry (CM)
- LSPE_ACCOUNT_LIST.qry (PE)
- LSPE_CHANNELHISTORY.qry (PE)
- LSPE_CHANNELHISTORY_0.qry (PE)
- LSPE_GET_METER_VALUES.qry (PE)

These records are installed with the Oracle Utilities Quotations Management database schema. These can also be installed manually using either Data Manager or the PLIMPORT, INTDIMP, RFIMPEXP, and LSTIMEXP command line programs. See the *Data Manager User's Guide* and the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about using these tools.

Chapter 4

Oracle Utilities Quotations Management Rules Language Schedules, Riders, and Lists

This chapter describes the Rules Language rate schedules, riders, and lists used by the Oracle Utilities Quotations Management application, including:

- **Rate Schedule/Rider Format**
- **Contract Rate Schedules**
- **Contract Riders**
- **Contract Lists**
- **Pricing Rate Schedules**
- **Pricing Riders**
- **Pricing Lists**

Important Note!

Do not delete any of the rate schedules, riders, and lists described in this chapter. Modifications made to any of the rate schedules, riders, and lists described in this chapter are not supported by Oracle Utilities. Changes made to these can result in errors when using Oracle Utilities Quotations Management.

Rate Schedule/Rider Format

Each of the rate schedules and riders descriptions in this chapter use the following format:

Description

A brief description of what the rate schedule/rider does

Rate Form Code/Name

The code/name of the rate schedule/rider, from the Rate Form table.

Input

The input parameters used by the rate schedule/rider.

Logic

A step-by-step description of what the rate schedule does.

Results

The results generated by the rate schedule/rider.

Contract Rate Schedules

This section describes rate schedules used in the creation and processing of contracts.

Important Note!

Do not delete any of the rate schedules described in this section. Modifications made to any of the rate schedules described in this section are not supported by Oracle Utilities. Changes made to these rate schedules can result in errors when using Oracle Utilities Quotations Management.

Create Contract

The Create Contract rate schedule creates a contract by using passed parameters, directly or as keys to other tables. The Contract Type code will be used to select predefined contract terms and document types to associate to the contract through the Contract Type table. The Counter Party ID or list will be used to build an account list, with each account added as a contract item. The rate schedule also saves a record to the Contract History table noting the action.

Rate Form Code/Name

LSCM_CREATE_CONTRACT

Input

The Create Contract rate schedule uses the following input parameters and required data:

- Contractee ID (CONTRACTEE_ID)
- Contract ID (CONTRACT_ID)
- Contract Description (CONTRACT_DESCRIPTION)
- Contract Type (CONTRACT_TYPE)
- Contract Category (CONTRACT_CATEGORY)
- Parent Contract ID (PARENT_CONTRACT_ID)
- Start Time (START_TIME)
- End Time (END_TIME)
- Counter Party
    - Customer or Account flag (CUSTOMER_ACCOUNT_FLG)
    - List (INPUT_LIST)
    - Parameter (INPUT_ID)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Calculate Terms Rider (LSCM_CALCULATE_TERMS_RDR)

Load Attributes Rider (LSCM_LOAD_ATTRIBUTES_RDR)

Load Item Attributes Rider (LSCM_LOAD_ITEM_ATTRIBUTES_RDR)

Required Lists

Get Accounts (LSCM_GET_ACCOUNTS)

Get Customer (LSCM_GET_CUSTOMER)

Get Contract Type Terms (LSCM_GET_CONTRACTTYPE_TERM)

Get Contract Type Documents (LSCM_GET_CONTRACTTYPE_DOC)

Logic

The Create Contract rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
   - Revision Number: 0
   - Contract External Status: ACTIVE
   - Contract Category Code: CONTRACT
   - Contract Status Code: INPROCESS
   - Contract Item Status Code: INPROCESS
   - Contract History Action Type: CREATECONTRACT
   - Contract History Details: Created a new contract
3. Create a record in the Contract Table.
4. If the Contract Category is OFFER, create offer attributes (including COGS, Margin, Rate, Max Quantity, Quantity, Probability, and Valid Until date)
5. Create records in the Contract Terms table
6. Calculate the terms associated with the contract. See **Calculate Terms Rider** on page 4-27 for more information. (Optional)
7. Create records in the Contract Documents table
8. Create records in the Contract Items table
9. If the Contract Category is OFFER, create contract item attributes, including forecast options and product options.
10. Create a record in the Contract History table noting the creation of the contract.

Results

The Create Contract rate schedule produces the following results:

Contract Table - New Record

Column	Value
UIDCONTRACT	Next Value
CONTRACTID	CONTRACT_ID (from Input)
REVISION	0
CONTRACTTYPECODE	CONTRACT_TYPE (from Input)
STARTTIME	START_TIME (from Input)
STOPTIME	STOP_TIME (from Input)
DESCRIPTION	DESCRIPTION (from Input)
CONTRACTCATEGORYCODE	CONTRACT_CATEGORY (from Input)
CONTRACTSTATUSCODE	INPROCESS
EXTCONTRACTSTATUSCODE	ACTIVE
SUBMITTIME	NULL
APPROVETIME	NULL
EXECUTETIME	NULL
TERMINATETIME	NULL
USERID	Current User
UIDLIST	Customer/Account List, if passed
UIDWQITEM	NULL
PROPERTIES	NULL
COGS	NULL*
MARGIN	NULL*
RATE	NULL*
MAXQUANTITY	NULL*
QUANTITY	NULL*
PROBABILITY	NULL*
OFFERVALIDTIME	NULL*
UIDCONTRACTEE	CONTRACTEE_ID (from Input)

* These values can be set by customizing the LSCM_LOAD_ATTRIBUTES and LSCM_LOAD_ITEM_ATTRIBUTES riders.

Contract Terms Table - New Records

For each term associated with the Contract's Contract Type, a record is created in the Contract Terms table with the following values:

Column	Value
UIDCONTRACT	New Contract UID from Contract Table
UIDTERM	UIDTERM from Contract Type Term Table
STARTTIME	*DEFAULTSTARTTIME from Contract Type Term Table.
STOPTIME	*DEFAULTSTOPTIME from Contract Type Term Table.
VAL	DEFAULTVAL from Contract Type Term Table
ISSTANDARD	**ISSTANDARD from Contract Type Term Table
ISREQUIRED	**ISREQUIRED from Contract Type Term Table)
ISCALCULATED	**ISCALCULATED from Contract Type Term Table

*If no Default Start/Stop Time is available, this defaults to the Contract Start/Stop Time.

**Null values are replaced with N (No).

Contract Documents Table - New Records

For each document associated with the Contract's Contract Type, a record is created in the Contract Documents table with the following values

Column	Value
UIDCONDOC	Next Value
UIDCONTRACT	New Contract UID from Contract Table
UIDDOCUMENT	NULL
UIDCONTYPDOCREL	UIDCONTYPDOCREL from Contract Type Document Relationship table
CREATETIME	Current DateTime
USERID	Current UserID
ISSTANDARD	*ISSTANDARD from Contract Type Document Relationship table
ISREQUIRED	*ISREQUIRED from Contract Type Document Relationship table

*Null values are replaced with N (No).

Contract Items Table - New Records

For each account associated with the Contract, a record is created in the Contract Items table with the following values:

Column	Value
UIDCONTRACTITEM	Next Value
UIDCONTRACT	New Contract UID from Contract Table
UIDACCOUNT	(from Input or calculated)
UIDCUSTOMER	NULL
UIDPRODUCT	NULL
STARTTIME	START_TIME (from Input)
STOPTIME	STOP_TIME (from Input)
CONTRACTSTATUSCODE	INPROCESS
PRODUCTSTATUS	NULL
COMMODITY	NULL*
COGS	NULL*
MARGIN	NULL*
RATE	NULL*
MAXQUANTITY	NULL*
QUANTITY	NULL*
PROPERTIES	NULL*
UIDCHANNELCUT	NULL*

* These values can be set by customizing the LSCM_LOAD_ATTRIBUTES and LSCM_LOAD_ITEM_ATTRIBUTES riders.

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	CREATECONTRACT
USERID	Current UserID
DETAILS	Create a new contract

Calculate Price

The Calculate Price rate schedule calculates values associated to a contract. This process is triggered when the user selects the **View Calculations** action from the **Actions** drop-down list on the **Contract** screen. This rate schedule uses the LSCM_CALCULATE_TERMS_RDR rider to perform the calculations, modifies the Contract Terms table with the changes, and saves a record to the Contract History table noting the action.

Rate Form Name

LSCM_CALCULATE_PRICE

Input

The Calculate Price rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Calculate Terms Rider (LSCM_CALCULATE_TERMS_RDR)

Required Lists

Get Contract Terms (LSCM_GET_CONTRACT)

Logic

The Calculate Price rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. If the contract status is INPROCESS, calculate the terms associated with the contract using the Calculate Terms and Calculate Price riders (see **Calculate Terms Rider** on page 4-27 and **Calculate Price Rider** on page 4-29 for more information).
3. Create a record in the Contract History table noting the creation of the contract (performed by the Calculate Terms rider).

Results

The Calculate Terms rate schedule produces the following results:

Contract Terms Table - Updated Records

For each term associated with the Contract, a record is updated in the Contract Terms table with the following values:

Column	Value
UIDCONTRACT	No Change
UIDTERM	No Change
STARTTIME	Calculated Value
STOPTIME	Calculated Value
VAL	Calculated Value
ISSTANDARD	No Change
ISREQUIRED	No Change
ISCALCULATED	No Change

Calculate Terms

The Calculate Terms rate schedule calculates the terms associated to a contract that are of type CALCULATED. This process is triggered when the user clicks the **Calculate Terms** option on the **Terms** tab of the **Contract** screen. To do this it calls the related riders (LSCM_CALCULATE_TERMS_RDR and LSCM_CALCULATE_TERMS_CUST_RDR) to perform the calculations, modifies the Contract Terms table with the changes, and saves a record to the Contract History table noting the action.

Rate Form Name

LSCM_CALCULATE_TERMS

Input

The Calculate Terms rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Calculate Terms Rider (LSCM_CALCULATE_TERMS_RDR)

Required Lists

Get Contract Terms (LSCM_GET_CONTRACT_TERM)

Logic

The Calculate Terms rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Calculate the terms associated with the contract using the Calculate Terms and Calculate Custom Terms riders (see **Calculate Terms Rider** on page 4-27 and **Calculate Custom Terms Rider** on page 4-29 for more information).
3. Create a record in the Contract History table noting the creation of the contract (performed by the Calculate Terms rider).

Results

The Calculate Terms rate schedule produces the following results:

Contract Terms Table - Updated Records

For each term associated with the Contract, a record is updated in the Contract Terms table with the following values:

Column	Value
UIDCONTRACT	No Change
UIDTERM	No Change
STARTTIME	Calculated Value
STOPTIME	Calculated Value
VAL	Calculated Value
ISSTANDARD	No Change
ISREQUIRED	No Change
ISCALCULATED	No Change

Share Item Terms

The Share Item Terms rate schedule shares contract item product terms from one contract item to other contract items within the same contract. This process is triggered when the user clicks the **Apply Product Terms** option on the **Contract Product Terms** window (access from the **Items** tab of the **Contract** screen). To do this it calls the LSCM_SHARE_ITEM_TERMS_RDR riders to perform the calculations.

Rate Form Name

LSCM_SHARE_ITEM_TERMS

Input

The Calculate Terms rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)
- Account ID (ACCOUNT_ID)

Required Riders

Share Item Terms Rider (LSCM_SHARE_ITEM_TERMS_RDR)

Logic

The Share Item Terms rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Share the item terms associated with the contract using the Share Item Terms rider (see **Share Item Terms Rider** on page 4-30 for more information).

Revise Contract

The Revise Contract rate schedule creates a full copy of the current contract, marks the original contract as Revised, and marks the new contract as In Process. This allows the copying of entire contract, while allowing custom modifications through the included Revise Contract rider.

Rate Form Name

LSCM_REVISE_CONTRACT

Input

The Revise Contract rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR

Required Lists

Get Contract (LSCM_GET_CONTRACT)

Get Contract Item (LSCM_GET_CONTRACT_ITEM)

Get Contract Parent (LSCM_GET_CONTRACT_PARENT)

Get Contract Terms (LSCM_ CONTRACT_TERM)

Get Contract Documents (LSCM_ CONTRACT_DOCS)

Logic

The Revise Contract rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Contract Status Code - Old: REVISED
    - Contract External Status Code - New: ACTIVE
    - Contract Status Code - New: INPROCESS
    - Contract History Action Type: REVISECONTRACT
    - Contract History Details: Revised an existing contract
3. Update the original contract's Revision Number
4. Create a record in the Contract History table noting the revision of the contract.
5. Reset the Contract History values as follows (for the new contract):
    - Contract History Action Type: CREATECONTRACT
    - Contract History Details: Created a new contract
6. Retrieve the original contract and increments its Revision Number
7. Copy the original contract and creates a new record in the Contract Table
8. If the Contract Category is OFFER, create offer attributes (including COGS, Margin, Rate, Max Quantity, Quantity, Probability, and Valid Until date)
9. Create new records in the Contract Terms table for the new contract
10. Create new records in the Contract Documents table for the new contract
11. Create new records in the Contract Items table for the new contract
12. If the Contract Category is OFFER, create contract item attributes, including forecast options and product options.
13. Create a record in the Contract History table noting the creation of the new contract.

Results

The Revise Contract rate schedule produces the following results:

Contract Table - Updated Record

Column	Value
UIDCONTRACT	No Change
CONTRACTID	No Change
REVISION	No Change
CONTRACTTYPECODE	No Change
STARTTIME	No Change
STOPTIME	No Change
DESCRIPTION	No Change
CONTRACTCATEGORYCODE	No Change
CONTRACTSTATUSCODE	REVISED
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	No Change
APPROVETIME	No Change
EXECUTETIME	No Change
TERMINATETIME	No Change
USERID	No Change
UIDLIST	No Change
UIDWQITEM	No Change
PROPERTIES	No Change
COGS	No Change*
MARGIN	No Change*
RATE	No Change*
MAXQUANTITY	No Change*
QUANTITY	No Change*
PROBABILITY	No Change*
OFFERVALIDTIME	No Change*
UIDCONTRACTEE	No Change

* These values can be set by customizing the LSCM_LOAD_ATTRIBUTES and LSCM_LOAD_ITEM_ATTRIBUTES riders.

Contract Items Table - Updated Records

For each account associated with the Contract, the record in the Contract Items table is updated with the following values:

Column	Value
UIDCONTRACTITEM	No Change
UIDCONTRACT	No Change
UIDACCOUNT	No Change
UIDCUSTOMER	No Change
UIDPRODUCT	No Change
STARTTIME	No Change
STOPTIME	No Change
CONTRACTSTATUSCODE	REVISED
PRODUCTSTATUS	No Change
COMMODITY	No Change
COGS	No Change*
MARGIN	No Change*
RATE	No Change*
MAXQUANTITY	No Change*
QUANTITY	No Change*
PROPERTIES	No Change*
UIDCHANNELCUT	No Change*

* These values can be set by customizing the LSCM_LOAD_ATTRIBUTES and LSCM_LOAD_ITEM_ATTRIBUTES riders.

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	REVISECONTRACT
USERID	Current UserID
DETAILS	Revised an existing contract

Contract Table - New Record

Column	Value
UIDCONTRACT	Next Value
CONTRACTID	CONTRACT_ID (from Input)
REVISION	REVISION (from Input) +1
CONTRACTTYPECODE	From Original Contract
STARTTIME	From Original Contract
STOPTIME	From Original Contract
DESCRIPTION	From Original Contract
CONTRACTCATEGORYCODE	From Original Contract
CONTRACTSTATUSCODE	INPROCESS
EXTCONTRACTSTATUSCODE	ACTIVE
SUBMITTIME	NULL
APPROVETIME	NULL
EXECUTETIME	NULL
TERMINATETIME	NULL
USERID	Current User
UIDLIST	From Original Contract
UIDWQITEM	NULL
PROPERTIES	From Original Contract
COGS	From Original Contract
MARGIN	From Original Contract
RATE	From Original Contract
MAXQUANTITY	From Original Contract
QUANTITY	From Original Contract
PROBABILITY	From Original Contract
OFFERVALIDTIME	From Original Contract
UIDCONTRACTEE	From Original Contract

Contract Terms Table - New Records

For each term associated with the original contract, a record is created in the Contract Terms table with the following values:

Column	Value
UIDCONTRACT	New Contract UID from Contract Table
UIDTERM	From Original Contract
STARTTIME	From Original Contract
STOPTIME	From Original Contract
VAL	From Original Contract
ISSTANDARD	From Original Contract
ISREQUIRED	From Original Contract
ISCALCULATED	From Original Contract

Contract Documents Table - New Records

For each document associated with the original contract, a record is created in the Contract Documents table with the following values

Column	Value
UIDCONDOC	New Contract UID from Contract Table
UIDCONTRACT	From Original Contract
UIDDOCUMENT	From Original Contract
UIDCONTYPDOCREL	From Original Contract
CREATETIME	Current DateTime
USERID	Current UserID
ISSTANDARD	From Original Contract
ISREQUIRED	From Original Contract

Contract Items Table - New Records

For each account associated with the original contract, a record is created in the Contract Items table with the following values:

Column	Value
UIDCONTRACTITEM	Next Value
UIDCONTRACT	New Contract UID from Contract Table
UIDACCOUNT	From Original Contract
UIDPRODUCT	From Original Contract
STARTTIME	From Original Contract
STOPTIME	From Original Contract
CONTRACTSTATUSCODE	INPROCESS

Column	Value
PRODUCTSTATUS	From Original Contract
COMMODITY	From Original Contract
COGS	From Original Contract
MARGIN	From Original Contract
RATE	From Original Contract
MAXQUANTITY	From Original Contract
QUANTITY	From Original Contract
PROPERTIES	From Original Contract
UIDCHANNELCUT	From Original Contract

Approve Contract

The Approve Contract rate schedule modifies an existing contract to associate a work queue approval process to the contract. The work queue approval process is calculated through a link from the contract type to a work queue type. A work queue item of this work queue type is created and associated to the contract.

Rate Form Name

LSCM_CONTRACT_APPROVALS

Input

The Approve Contract rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Required Lists

Get Contract (LSCM_GET_CONTRACT)

Get WQ Type (LSCM_GET_WQTYPE)

Logic

The Approve Contract rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Contract Status Code - New: INAPPROVAL
    - Contract History Action Type: INAPPROVALCONTRACT
    - Contract History Details: Set Contract Status to INAPPROVAL
3. Create an open work queue item based on the work queue type associated with the Contract Type
    a. Define work queue item XML structure

b. Create open work queue item

b. Retrieve XML document containing work queue item

b. Extract UIDWQITEM to be inserted into Contract Table

3. Update the contract record in the Contract Table, including inserting the created work queue item (UIDWQITEM).
4. Create a record in the Contract History table noting the change of the contract status to INAPPROVAL.

Results

The Approve Contract rate schedule produces the following results:

Contract Table - Updated Record

Column	Value
UIDCONTRACT	No Change
CONTRACTID	No Change
REVISION	No Change
CONTRACTTYPECODE	No Change
STARTTIME	No Change
STOPTIME	No Change
DESCRIPTION	No Change
CONTRACTCATEGORYCODE	No Change
CONTRACTSTATUSCODE	INAPPROVAL
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	No Change
APPROVETIME	No Change
EXECUTETIME	No Change
TERMINATETIME	No Change
USERID	No Change
UIDLIST	No Change
UIDWQITEM	UIDWQITEM (calculated)
PROPERTIES	No Change
COGS	No Change
MARGIN	No Change
RATE	No Change
MAXQUANTITY	No Change
QUANTITY	No Change
PROBABILITY	No Change

Column	Value
OFFERVALIDTIME	No Change
UIDCONTRACTEE	No Change

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	INAPPROVALCONTRACT
USERID	Current UserID
DETAILS	Set Contract Status to INAPPROVAL

Submit Contract

The Submit Contract rate schedule calls the Submit Contract rider to perform any customized business logic, and changes the status of the contract to SUBMITTED.

Rate Form Name

LSCM_SUBMIT_CONTRACT

Input

The Submit Contract rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Logic

The Submit Contract rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
   - Contract Status Code - New: SUBMITTED
   - Contract History Action Type: SUBMITTEDCONTRACT
   - Contract History Details: Set Contract Status to SUBMITTED
   - APPROVETIME: Null
3. Perform customized business logic configured in the Submit Contract Rider (Optional).
4. Update the contract record in the Contract Table.
5. Create a record in the Contract History table noting the change of the contract status to SUBMITTED.

Results

The Submit Contract rate schedule produces the following results:

Contract Table - Updated Record

Column	Value
UIDCONTRACT	No Change
CONTRACTID	No Change
REVISION	No Change
CONTRACTTYPECODE	No Change
STARTTIME	No Change
STOPTIME	No Change
DESCRIPTION	No Change
CONTRACTCATEGORYCODE	No Change
CONTRACTSTATUSCODE	SUBMITTED
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	Current DateTime
APPROVETIME*	Null
EXECUTETIME	No Change
TERMINATETIME	No Change
USERID	No Change
UIDLIST	No Change
UIDWQITEM	No Change
PROPERTIES	No Change
COGS	No Change
MARGIN	No Change
RATE	No Change
MAXQUANTITY	No Change
QUANTITY	No Change
PROBABILITY	No Change
OFFERVALIDTIME	No Change
UIDCONTRACTEE	No Change

*The APPROVETIME variable in the rate schedule column defaults to Null and is used to fill the APPROVETIME column. The Submit Contract rider can be used to modify this variable and the column value.

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	SUBMITTEDCONTRACT
USERID	Current UserID
DETAILS	Set Contract Status to SUBMITTED

Execute Contract

The Execute Contract rate schedule calls the Execute Contract rider to perform any customized business logic, and changes the status of the contract to EXECUTED.

Rate Form Name

LSCM_EXECUTE_CONTRACT

Input

The Execute Contract rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Logic

The Execute Contract rate schedule performs the following steps:

1. Sets Report Parameters and Labels
2. Sets default values as follows:
    - Contract Status Code - New: EXECUTED
    - Contract History Action Type: EXECUTEDCONTRACT
    - Contract History Details: Set Contract Status to EXECUTED
3. Performs customized business logic configured in the Execute Contract rider (Optional).
4. Updates the contract record in the Contract Table.
5. Creates a record in the Contract History table noting the change of the contract status to EXECUTED.

Results

The Execute Contract rate schedule produces the following results:

Contract Table - Updated Record

Column	Value
UIDCONTRACT	No Change
CONTRACTID	No Change
REVISION	No Change
CONTRACTTYPECODE	No Change
STARTTIME	No Change
STOPTIME	No Change
DESCRIPTION	No Change
CONTRACTCATEGORYCODE	No Change
CONTRACTSTATUSCODE	EXECUTED

Column	Value
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	No Change
APPROVETIME	No Change
EXECUTETIME	Current DateTime
TERMINATETIME	No Change
USERID	No Change
UIDLIST	No Change
UIDWQITEM	No Change
PROPERTIES	No Change
COGS	No Change
MARGIN	No Change
RATE	No Change
MAXQUANTITY	No Change
QUANTITY	No Change
PROBABILITY	No Change
OFFERVALIDTIME	No Change
UIDCONTRACTEE	No Change

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	EXECUTEDCONTRACT
USERID	Current UserID
DETAILS	Set Contract Status to EXECUTED

Terminate Contract

The Terminate Contract rate schedule calls the Terminate Contract rider to perform any customized business logic, and changes the status of the contract to TERMINATED.

Rate Form Name

LSCM_TERMINATE_CONTRACT

Input

The Terminate Contract rate schedule uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)

- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Logic

The Terminate Contract rate schedule performs the following steps:

1. Sets Report Parameters and Labels
2. Sets default values as follows:
   - Contract Status Code - New: TERMINATED
   - Contract History Action Type: TERMINATEDCONTRACT
   - Contract History Details: Set Contract Status to TERMINATED
3. Performs customized business logic configured in the Terminate Contract rider (Optional).
4. Updates the contract record in the Contract Table.
5. Creates a record in the Contract History table noting the change of the contract status to TERMINATED.

Results

The Terminate Contract rate schedule produces the following results:

Contract Table - Updated Record

Column	Value
UIDCONTRACT	No Change
CONTRACTID	No Change
REVISION	No Change
CONTRACTTYPECODE	No Change
STARTTIME	No Change
STOPTIME	No Change
DESCRIPTION	No Change
CONTRACTCATEGORYCODE	No Change
CONTRACTSTATUSCODE	TERMINATED
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	No Change
APPROVETIME	No Change
EXECUTETIME	No Change
TERMINATETIME	Current DateTime
USERID	No Change
UIDLIST	No Change
UIDWQITEM	No Change

Column	Value
PROPERTIES	No Change
COGS	No Change
MARGIN	No Change
RATE	No Change
MAXQUANTITY	No Change
QUANTITY	No Change
PROBABILITY	No Change
OFFERVALIDTIME	No Change
UIDCONTRACTEE	No Change

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	TERMINATEDCONTRACT
USERID	Current UserID
DETAILS	Set Contract Status to TERMINATED

Contract Riders

This section describes Rules Language riders used in the creation and processing of contracts.

Add History Rider

The Add History rider saves a record to the Contract History table whenever an action is performed on the contract. This rider is included in all Oracle Utilities Quotations Management rate schedules.

Important Note!

Do not delete the Add History rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_ADD_HISTORY_RDR

Input

The Add History uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)
- Contract History Action Type (HISTACTIONTYPE)
- Contract History Details (HISTDETAILS)

These input values come from the specific rate schedule that call the rider.

Logic

The Add History rider creates a record in the Contract History table based on the Contract and Action Type.

Results

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	HISTACTIONTYPE (from Input)
USERID	Current UserID
DETAILS	HISTDETAILS (from Input)

Calculate Terms Rider

The Calculate Terms rider calculates the terms associated to a contract that are of type CALCULATED, calling the Calculate Customer Terms rider to perform the calculations, and modifies records in the Contract Terms table with the changes.

Important Note!

Do not delete the Calculate Terms rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_CALCULATE_TERMS_RDR

Input

The Calculate Terms rider uses the following input parameters and required data:

- Contract ID (CONTRACT_ID)
- Contract Revision (REVISION)

Required Riders

Add History Rider (LSCM_ADD_HISTORY_RDR)

Calculate Custom Terms Rider (LSCM_CALCULATE_TERMS_CUST_RDR)

Required Lists

Get Contract Terms (LSCM_GET_CONTRACT_TERM)

Logic

The Calculate Terms rider performs the following steps:

1. Sets Report Parameters and Labels
2. Sets default values as follows:
    - Contract History Action Type: CALCULATETERMS
    - Contract History Details: Calculated terms for this contract
3. Calculates the terms associated with the contract
4. Updates the records in the Contract Terms table
5. Calls the Calculate Custom Terms rider. See **Calculate Custom Terms Rider** on page 4-29 for more information. (Optional)
6. Creates a record in the Contract History table noting the creation of the contract.

Results

The Calculate Terms rider produces the following results:

Contract Terms Table - Updated Records

For each term associated with the Contract, a record is updated in the Contract Terms table with the following values:

Column	Value
UIDCONTRACT	No Change
UIDTERM	No Change
STARTTIME	Calculated Value
STOPTIME	Calculated Value
VAL	Calculated Value
ISSTANDARD	No Change
ISREQUIRED	No Change
ISCALCULATED	No Change

Contract History Table - New Record

Column	Value
UIDCONTRACT	CONTRACT_ID/REVISION (from Input)
ACTIONTIME	Current DateTime
ACTIONTYPE	CALCULATETERMS
USERID	Current UserID
DETAILS	Calculated terms for this contract

Load Attributes Rider

The Load Attributes rider creates attributes used by pricing contracts. It is provided to allow for customized business logic to be called from the Create Contract and Revise Contract rate schedules. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Important Note!

Do not delete the Load Attributes rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_LOAD_ATTRIBUTES_RDR

Load Item Attributes Rider

The Load Item Attributes rider creates contract item attributes used by pricing contracts. It is provided to allow for customized business logic to be called from the Create Contract and Revise Contract rate schedules. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Important Note!

Do not delete the Load Item Attributes rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_LOAD_ITEM_ATTRIBUTES_RDR

Calculate Custom Terms Rider

The Calculate Custom Terms rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Calculate Terms rider. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSCM_CALCULATE_TERMS_CUST_RDR

Calculate Price Rider

The Calculate Price rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Calculate Price rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Important Note!

Do not delete the Calculate Price rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_CALCULATE_PRICE_RDR

Share Item Terms Rider

The Share Item Terms rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Share Item Terms rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Important Note!

Do not delete the Share Item Terms rider. **Modifications made to this rider are not supported by Oracle Utilities.** Changes made to this rider can result in errors when using Oracle Utilities Quotations Management.

Rate Form Name

LSCM_SHARE_ITEM_TERMS_RDR

Submit Contract Rider

The Submit Contract rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Submit Contract rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSCM_SUBMIT_CONTRACT_RDR

Execute Contract Rider

The Execute Contract rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Execute Contract rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSCM_EXECUTE_CONTRACT_RDR

Terminate Contract Rider

The Terminate Contract rider is an empty rider that contains no Rules Language logic. It is provided to allow for customized business logic to be called from the Terminate Contract rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSCM_TERMINATE_CONTRACT_RDR

Contract Lists

The contract rate schedules and riders use a number of pre-defined Table-Column lists to retrieve the appropriate records from the Oracle Utilities Data Repository for each process.

Important Note!

Do not delete or modify any of the lists described in this section in any way. Changes made to these lists can result in errors when using Oracle Utilities Quotations Management.

List Format

Each of the list descriptions in this chapter uses the following format:

Description

A brief description of the list.

List Name

The name of the list.

Parameters

The parameters upon which the list is based. This includes a database field and a Rules Language Identifier for each parameter.

Table Accessed

The table (and column where applicable) in the Oracle Utilities Data Repository from which the retrieved records are taken.

Path

The path between tables in the Oracle Utilities Data Repository used to retrieve the records. This is only provided when the path includes more than one table.

Get Accounts

The Get Accounts lists retrieves all of the Account records related to a specified Customer.

List Name

LSCM_GET_ACCOUNTS

Parameters

The Get Accounts list uses the following parameters:

Database Field	Identifier
Customer ID	CUSTOMERID

Table Accessed

The Get Accounts list accesses the Account table.

Path

Customer > Account

Get Customer

The Get Customer list retrieves the Customer ID related to a specified Account.

List Name

LSCM_GET_CUSTOMER

Parameters

The Get Customer list uses the following parameters:

Database Field	Identifier
Account ID	ACCOUNTID

Table Accessed

The Get Customer list accesses the Customers table.

Path

Account > Customer

Get Contract

The Get Contract list retrieves the contract record for a specified contract.

List Name

LSCM_GET_CONTRACT

Parameters

The Get Contract list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract list accesses the Contract table.

Get Parent Contract

The Get Parent Contract list retrieves the parent contract record for a specified contract.

List Name

LSCM_GET_CONTRACT_PARENT

Parameters

The Get Parent Contract list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract list accesses the Contract Relationship table.

Get Child Contract

The Get Child Contract list retrieves the child contract record for a specified contract.

List Name

LSCM_GET_CONTRACT_CHILD

Parameters

The Get Child Contract list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract list accesses the Contract Relationship table.

Get Contract Items

The Get Contract Items list retrieves Contract Item records for a specified contract

List Name

LSCM_GET_CONTRACT_ITEM

Parameters

The Get Contract Items list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract Items list accesses the Contract Item table.

Path

Contract > Contract Item

Get Contract Terms

The Get Contract Terms list retrieves Contract Term records for a specified contract

List Name

LSCM_ GET_CONTRACT_TERM

Parameters

The Get Contract Terms list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract Terms list accesses the Contract Terms table.

Path

Contract > Contract Term

Get Contract Documents

The Get Contract documents list retrieves Contract Document records for a specified contract

List Name

LSCM_GET_CONTRACT_DOC

Parameters

The Get Contract Documents list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract Documents list accesses the Contract Document table.

Path

Contract > Contract Document

Get Contract Type Terms

The Get Contract Type Terms list retrieves Contract Term records for a specified Contract Type.

List Name

LSCM_GET_CONTRACTTYPE_TERM

Parameters

The Get Contract Type Terms list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract Type Terms list accesses the Contract Type Terms table.

Path

Contract > Contract Type > Contract Type Term

Get Contract Type Documents

The Get Contract Type Documents list retrieves Contract Document records for a specified Contract Type.

List Name

LSCM_GET_CONTRACTTYPE_DOC

Parameters

The Get Contract Type Documents list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Contract Type Documents list accesses the Contract Type Document Relationship table.

Path

Contract > Contract Type > Contract Type Document Relationship

Get Last Contract Revision

The Get Last Contract Revision list retrieves the last (highest) revision for a specified contract.

List Name

LSCM_LAST_CONTRACT_REVISION

Parameters

The Get Last Contract Revision list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Last Contract Revision list accesses Revision column in the Contract table.

Get Work Queue Type

The Get Work Queue Type list retrieves the Work Queue Type Code related to a specified contract based on the Contract Type.

List Name

LSCM_GET_WQTYPE

Parameters

The Get Work Queue Type list uses the following parameters:

Database Field	Identifier
Contract ID	CONTRACTID
Revision	REVISION

Table Accessed

The Get Work Queue Type list accesses Work Queue Type Code column in the Work Queue Type table.

Path

Contract > Contract Type > Work Queue Type

Pricing Rate Schedules

This section describes rate schedules used in pricing processing, including validating, estimating, and normalizing usage data, and validating and finalizing forecasts.

Important Note!

Do not delete any of the rate schedules described in this section. Modifications made to any of the rate schedules described in this section are not supported by Oracle Utilities. Changes made to these rate schedules can result in errors when using Oracle Utilities Quotations Management.

Validate Usage (Standard)

The Validate Usage (Standard) rate schedule reviews usage for a list of accounts, and mark each point of usage as valid or invalid.

Rate Form Code/Name

LSPE_VAL_USAGE

Input

The Validate Usage (Standard) rate schedule takes the following input parameters and required data:

- Account List
- Account ID
- Validation Start Date
- Validation End Date

Required Riders

Validate Usage Rider (LSPE_VAL_USAGE_RDR)

Required Lists

Pricing Account List (PRICING_ACCOUNT_LIST)

Required Data

Interval Data for DEFAULT_DATA,1

Logic

The Validate Usage (Standard) rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Status Code: FAILED_VALIDATION
    - Channel History Data Type: I
    - Channel History Billed: Y
    - Channel History Totalize: T
    - Channel History End Use Code: PE
    - Channel History Usage Data: RAW
    - Validation Method: STANDARD
    - Validation Message: " "

- Raw Channel Number (RAW_CHANNELNUM): 100
- Validated Channel Number (VAL_CHANNELNUM): 200

3. Load default interval data (DEFAULT_DATA,1) for the validation period.
4. For each account, perform validation logic in the Validate Usage Rider.
5. Create a new record in the Channel History table for this account, with usage data 'RAW'.
6. Create a new record in the Recorder, Channel, Channel Cut Header and related tables for the raw interval data.
7. Create a new record (or update an existing record ia appropriate) in the Channel History table for this account, with usage data 'VALIDATED'.
8. Create a new record (or update an existing record if appropriate) in the Recorder, Channel, Channel Cut Header, and related tables for the validated interval data.

Results

The Validate Usage (Standard) rate schedule creates two records in the Channel and Channel History tables for each account in the account list, one for Raw data, one for Validated Data. The specifics values for each record are as follows:

Raw Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	100 (RAW)

Raw Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Usage Start parameter
STOPTIME	Per Usage Stop parameter
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	RAW
STATUSCODE	NOT VALIDATED
VALIDATIONMETHOD	null
VALIDATIONMSG	null
ESTIMATIONREQUIRED	null

Validated Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	200 (VALIDATED)

Validated Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Usage Start parameter
STOPTIME	Per Usage Stop parameter
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	RAW
STATUSCODE	VALIDATED or VALIDATION_FAILED as appropriate
VALIDATIONMETHOD	STANDARD
VALIDATIONMSG	Based on validation logic
ESTIMATIONREQUIRED	null

In addition, the Validate Usage (Standard) rate schedule creates related records in the Recorder, Channel Cut Header, Channel Cut Data, and Channel Cut Validation Message tables.

Validate Usage (Loose)

The Validate Usage (Loose) rate schedule reviews usage for a list of accounts, and mark each point of usage as valid or invalid. This rate schedule is used only if usage for an account or accounts fails the standard validation.

Rate Form Code/Name

LSPE_VAL_USAGE_LOOSE

Input

The Validate Usage (Loose) rate schedule takes the following input parameters and required data:

- Account List
- Account ID
- Validation Start Date
- Validation End Date

Required Riders

Validate Usage Loose Rider (LSPE_VAL_USAGE_LOOSE_RDR)

Required Lists

Pricing Account List (PRICING_ACCOUNT_LIST)

Required Data

Interval Data for DEFAULT_DATA,1

Logic

The Validate Usage (Loose) rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Status Code: FAILED_VALIDATION
    - Channel History Data Type: I
    - Channel History Billed: Y
    - Channel History Totalize: T
    - Channel History End Use Code: PE
    - Channel History Usage Data: RAW
    - Validation Method: STANDARD
    - Validation Message: " "
    - Raw Channel Number (RAW_CHANNELNUM): 100
    - Validated Channel Number (VAL_CHANNELNUM): 200
3. Load default interval data (DEFAULT_DATA,1) for the validation period.
4. For each account, perform validation logic in the Validate Usage Loose Rider.
5. Create a new record in the Channel History table for this account, with usage data 'RAW'.
6. Create a new record in the Recorder, Channel, Channel Cut Header and related tables for the raw interval data.

7. Create a new record (or update an existing record ia appropriate) in the Channel History table for this account, with usage data 'VALIDATED'.
8. Create a new record (or update an existing record if appropriate) in the Recorder, Channel, Channel Cut Header, and related tables for the validated interval data.

Results

The Validate Usage (Loose) rate schedule creates two records in the Channel and Channel History tables for each account in the account list, one for Raw data, one for Validated Data. The specifics values for each record are as follows:

Raw Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	100 (RAW)

Raw Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Usage Start parameter
STOPTIME	Per Usage Stop parameter
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	RAW
STATUSCODE	NOT VALIDATED
VALIDATIONMETHOD	null
VALIDATIONMSG	null
ESTIMATIONREQUIRED	null

Validated Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	200 (VALIDATED)

Validated Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Usage Start parameter
STOPTIME	Per Usage Stop parameter

Column	Value
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	RAW
STATUSCODE	VALIDATED or VALIDATION_FAILED as appropriate
VALIDATIONMETHOD	STANDARD
VALIDATIONMSG	Based on validation logic
ESTIMATIONREQUIRED	null

In addition, the Validate Usage (Loose) rate schedule creates related records in the Recorder, Channel Cut Header, Channel Cut Data, and Channel Cut Validation Message tables.

Estimate and Normalize Usage

The Estimate and Normalize Usage rate schedule reviews usage for a list of accounts, and creates estimated and normalized interval data.

Rate Form Code/Name

LSPE_EST_USAGE

Input

The Estimate and Normalize Usage rate schedule takes the following input parameters and required data:

- Account List
- Account ID
- Estimation Start Date
- Estimation End Date

Required Riders

Estimate Usage Rider (LSPE_EST_USAGE_RDR)

Normalize Usage Rider (LSPE_NORM_USAGE_RDR)

Required Lists

Channel History List (LSPE_CHANNELHISTORY)

Required Data

Interval Data for DEFAULT_DATA,1

Logic

The Estimate and Normalize Usage rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Validated Channel Number (VAL_CHANNELNUM): 200
    - Estimated Channel Number (EST_CHANNELNUM): 300

- Normalized Channel Number (NORM_CHANNELNUM): 400
- Estimated Status Code: FAILED_ESTIMATION
- Normalization Status Code: FAILED_NORMALIZATION
- Estimation Required Flag: Y

3. Load default interval data (DEFAULT_DATA,1) for the estimation period.
4. For each account, perform estimation logic in the Estimate Usage Rider.
5. Create records in the Recorder, Channel, and Channel History table for the estimated data.
6. *Optional*. For each account, perform normalization logic from the Normalize Usage rider.
7. Create records in the Recorder, Channel, and Channel History table for the estimated data.

Results

The Estimate and Normalize Usage rate schedule creates two records in the Channel and Channel History tables for each account in the account list, one for Estimated data, one for Normalized Data. The specifics values for each record are as follows:

Estimated Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	300 (ESTIMATED)

Estimated Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Estimation Start parameter
STOPTIME	Per Estimation Stop parameter
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	Per previous validated data
STATUSCODE	ESTIMATED
VALIDATIONMETHOD	Per previous validated data
VALIDATIONMSG	Per previous validated data
ESTIMATIONREQUIRED	Y

Normalized (RTQ) Data (Channel Table)

Column	Value
RECORDER	Account ID
CHANNEL	400 (NORMALIZED)

Normalized (RTQ) Data (Channel History)

Column	Value
UIDACCOUNT	Per account list/account parameter
STARTTIME	Per Estimation Start parameter
STOPTIME	Per Estimation Stop parameter
ENDUSECODE	PE
UOMCODE	Per Bill Determinant Code from Meter Value record
USAGEDATA	Per previous validated data
STATUSCODE	RTQ
VALIDATIONMETHOD	Per previous validated data
VALIDATIONMSG	Per previous validated data
ESTIMATIONREQUIRED	Per previous validated data

In addition, the Estimate and Normalize Usage rate schedule creates related records in the Recorder, Channel Cut Header, Channel Cut Data, and Channel Cut Validation Message tables.

Validate Forecast

The Validate Forecast rate schedule reviews and validates forecast data for a list of accounts.

Rate Form Code/Name

LSUE_VAL_FORECAST

Input

The Validate Forecast rate schedule takes the following input parameters and required data:

- Forecast ID

Required Riders

Validate Forecast Rider (LSUE_VAL_FORECAST_RDR)

Required Lists

Get Forecast List (LSUE_GET_FORECAST) (used by the Validate Forecast Rider)

Required Data

Interval Data for DEFAULT_DATA,1 (used by the Validate Forecast Rider)

Logic

The Validate Forecast rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
   - Status Code: SUCCESSFUL
3. Call the Validate Forecast Rider (LSUE_VAL_FORECAST_RDR), and perform validation logic.

Results

Output from the Validate Forecast rate schedule is created by the Validate Forecast rider (LSUE_VAL_FORECAST_RDR).

Finalize Forecast

The Finalize Forecast rate schedule reviews forecast data for a list of accounts, and sets the status code for the forecast to final.

Rate Form Code/Name

LSUE_FINALIZE_FORECAST

Input

The Finalize Forecast rate schedule takes the following input parameters and required data:

- Forecast ID

Required Riders

Validate Forecast Rider (LSUE_VAL_FORECAST_RDR)

Finalize Forecast Rider (LSUE_FINALIZE_FORECAST_RDR)

Required Lists

Get Forecast List (LSUE_GET_FORECAST) (used by the Validate Forecast Rider)

Required Data

Interval Data for DEFAULT_DATA,1 (used by the Validate Forecast Rider)

Logic

The Finalize Forecast rate schedule performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Status Code: FINAL
3. Call the Validate Forecast Rider (LSUE_VAL_FORECAST_RDR) and perform validation logic.
4. Call the Finalize Forecast Rider (LSUE_FINALIZE_FORECAST_RDR), and perform finalize logic.
5. If the validation (performed by the Validate Forecast Rider) was successful, update the status of the forecast to FINAL.

Results

Output from the Finalize Forecast rate schedule is created by the Validate Forecast rider (LSUE_VAL_FORECAST_RDR).

Pricing Riders

This section describes Rules Language riders used in pricing processing, including validating, estimating, and normalizing usage data, and validating and finalizing forecasts.

Important Note!

Do not delete any of the riders described in this section. **Modifications made to these riders are not supported by Oracle Utilities.** Changes made to these riders can result in errors when using Oracle Utilities Quotations Management.

Validate Usage (Standard) Rider

The Validate Usage (Standard) rider performs data validation for usage data used in pricing contracts. It is provided to allow for customized business logic to be called from the Validate Usage (Standard) rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSUE_VAL_USAGE_RDR

Validate Usage (Loose) Rider

The Validate Usage (Loose) rider performs data validation for usage data used in pricing contracts. It is provided to allow for customized business logic to be called from the Validate Usage (Loose) rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSUE_VAL_USAGE_LOOSE_RDR

Estimate Usage Rider

The Estimate Usage rider performs data estimation for usage data used in pricing contracts. It is provided to allow for customized business logic to be called from the Estimate Usage rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSUE_EST_USAGE_RDR

Results

The Estimate Usage rider sets the following values for estimated data:

- Status Code: ESTIMATED
- Estimation Required: Y

Normalize Usage Rider

The Normalize Usage rider performs weather normalization for usage data used in pricing contracts. Weather normalization is performed after estimation. This rider is provided to allow for customized business logic to be called from the Normalize Usage rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSUE_NORM_USAGE_RDR

Results

The Normalize Usage rider sets the following values for estimated and normalized data:

- Status Code: RTQ

Validate Forecast

The Validate Forecast rider performs validations for forecasts for a list of accounts.

Rate Form Code/Name

LSUE_VAL_FORECAST_RDR

Input

The Validate Forecast rider takes the following input parameters and required data:

- Forecast ID

Required Lists

Get Forecast List (LSUE_GET_FORECAST)

Required Data

Interval Data for DEFAULT_DATA,1

Logic

The Validate Forecast rider performs the following steps:

1. Set Report Parameters and Labels
2. Set default values as follows:
    - Channel History Channel Number (CH_CHANNELNUM): 1000
    - Channel History Usage Data (CH_USAGEDATA): FORECAST
    - Channel History End Use Code (CH_ENDUSECODE): FORECAST
    - Channel History UOM Code (CH_UOMCODE): 01
    - Channel History Data Type (CH_DATATYPE): I
    - Channel History Totalize (CH_TOTALIZE): T
    - Channel History Billed (CH_BILLED): N
    - Validation Success (VALIDATION_SUCCESS): FALSE
3. Get forecast data (from Get Forecast List)
4. Load default interval data (DEFAULT_DATA,1) for the forecast period.
5. For each account, perform custom forecast logic.

6. Create a new record in the Recorder, Channel, Channel Cut Header and related tables for the forecast data (now stored as interval data).
7. Create a new record (or update an existing record ia appropriate) in the Channel History table for this account, with usage data 'FORECAST'.

Results

The Validate Forecast rider creates records in the Channel and Channel History tables for each account in the account list. The specifics values for each record are as follows:

Channel Table

Column	Value
RECORDER	Forecast ID
CHANNEL	1000 (CH_CHANNELNUM)

Channel History

Column	Value
RECORDER	Forecast ID (from RECORDER table)
CHANNEL	1000 (CH_CHANNELNUM)
STARTTIME	Forecast Start Time
STOPTIME	Forecast Stop Time
ACCOUNTID	Account ID (from Forecast)
USAGEDATA	FORECAST (CH_USAGEDATA)
ENDUSECODE	FORECAST (CH_ENDUSECODE)
STATUSCODE	Status Code from Validate Forecast Rate Schedule
UOMCODE	“01” (CH_UOMCODE)
DATETYPE	“I” (CH_DATATYPE)
TOTALIZED	“T” (CH_TOTALIZE)
BILLED	“N” (CH_BILLED)
FORECASTID	Forecast ID

In addition, the Validate Forecast rider creates related records in the Recorder, Channel Cut Header, Channel Cut Data, and Channel Cut Validation Message tables.

Finalize Forecast Rider

The Finalize Forecast rider performs custom logic for finalizing forecast data. Finalization is performed after validation. This rider is provided to allow for customized business logic to be called from the Finalize Forecast rate schedule. See **Chapter 7**: **Adding Customized Business Logic to Oracle Utilities Quotations Management** for more information.

Rate Form Name

LSUE_FINALIZE_FORECAST_RDR

Pricing Lists

The pricing rate schedules and riders use a number of pre-defined Table-Column lists to retrieve the appropriate records from the Oracle Utilities Data Repository for each process.

Important Note!

Do not delete or modify any of the lists described in this section in any way. Changes made to these lists can result in errors when using Oracle Utilities Quotations Management.

List Format

Each of the list descriptions in this chapter uses the following format:

Description

A brief description of the list.

List Name

The name of the list.

Parameters

The parameters upon which the list is based. This includes a database field and a Rules Language Identifier for each parameter.

Table Accessed

The table (and column where applicable) in the Oracle Utilities Data Repository from which the retrieved records are taken.

Path

The path between tables in the Oracle Utilities Data Repository used to retrieve the records. This is only provided when the path includes more than one table.

Pricing Account List

The Pricing Account List retrieves Account records related to a specified Account ID.

List Name

PRICING_ACCOUNT_LITS

Parameters

The Pricing Account List list uses the following parameters:

Database Field	Identifier
Account ID	ACCOUNTID

Table Accessed

The Pricing Account List accesses the Account table.

Channel History

The Channel History list retrieves Channel History records based on a specified Account ID, Recorder, Channel, Start Time, and Stop Time.

List Name

LSPE_CHANNELHISTORY

Parameters

The Channel History list uses the following parameters:

Database Field	Identifier
Account ID	ACCOUNTID
Channel Number	CHANNELNUM
Recorder ID	RECORDERID
Start Time	STARTTIME
Stop Time	STOPTIME

Table Accessed

The Channel History list accesses the Channel History table.

Path

Account > Recorders > Channel > Channel History

Get Forecast List

The Get Forecast List retrieves Forecast records related to a specified Forecast ID.

List Name

LSUE_GET_FORECAST

Parameters

The Get Forecast List list uses the following parameters:

Database Field	Identifier
Forecast ID	FORECAST_ID

Table Accessed

The Get Forecast List accesses the Forecast table.

Chapter 5

Setting Up Contract Database Tables

This chapter describes how to set up records in a number of tables in the Oracle Utilities Data Repository used by Oracle Utilities Quotations Management, including:

- **Setting Up Contractee Tables**
- **Setting Up Document Tables**
- **Setting Up Other Tables**

You set up these records in the Oracle Utilities Data Repository using the Data Manager application.

A Note about Predefined Records:

Several of the tables described in this chapter contain predefined records used by the Oracle Utilities Quotations Management application. These records are listed in the appropriate table description. **Do not modify or delete any of these predefined records**.

Setting Up Contractee Tables

Contractees are individuals within a business who can create contracts to be signed with counter parties (parties outside the business). Setting up contractees involves creating records in the following tables in the Oracle Utilities Data Repository:

- **Contractee Type Table**
- **Contractee Status Table**
- **Contractee Table**
- **Contractee Directory Table**

Contractee Type Table

Records in the Contractee Type table represent specific types of contractees that can create contracts. Records in this table include the following information:

- **Contractee Type Code**: A unique code that identifies the contractee type.
- **Description**: A description of the contractee type.

The Contractee Type table includes the following predefined records:

Code	Description
STANDARD	Standard

Contractee Status Table

Records in the Contractee Status table represent specific statuses that can be associated to specific contractees in the database. Records in this table include the following information:

- **Contractee Status Code**: A unique code that identifies the contractee status.
- **Description**: A description of the contractee status.

The Contractee Status table includes the following predefined records:

Code	Description
ACTIVE	Active
INACTIVE	Inactive

Contractee Table

Records in the Contractee table represent individual contractees in the database. Records in this table include the following information:

- **Contractee ID**: A unique ID for the contractee.
- **Name**: The name of the contractee.
- **Contractee Status**: The current status of the contractee, from the **Contractee Status Table**.
- **Contractee Type**: The current status of the contractee, from the **Contractee Type Table**.
- **Start Time**: An optional start time for the contractee in the database. This is the time after which the contractee is capable of creating contracts.
- **Stop Time**: An optional stop time for the contractee in the database. This is the time before which the contractee is capable of creating contracts.

Contractee Directory Table

Records in the Contractee Directory table contain directory information (such as phone number, address, etc.) for contractees in the database. Records in this table include the following information:

- **Contractee**: The contractee's ID.
- **Directory**: The directory that contains contact information (phone numbers, email address, etc.) for the contractee, from the **Directory** table.
- **Contact Type**: The contact type for the contractee.
- **Fax Flag**: A flag that indicates if results of calculations should be FAXED to the contractee.

Setting Up Document Tables

Documents refer to specific types of documents that can be related to a contract using the Oracle Utilities Quotations Management application. These might include a letter of credit, a proposal, or other documents. Setting up documents involves creating records in the following tables in the Oracle Utilities Data Repository:

- **Document Category Table**
- **Document Status Table**
- **Document Type Table**

Document Category Table

Records in the Document Category table represent specific categories of documents, such as contracts, proposals, etc,. Records in this table include the following information:

- **Document Category Code**: A unique code that identifies the document category.
- **Description**: A description of the document category.

The Document Category table includes the following predefined records:

Code	Description
CONTRACT	Contract

Document Status Table

Records in the Document Status table represent specific statuses that can be associated to specific documents in the database. Records in this table include the following information:

- **Document Status Code**: A unique code that identifies the document status.
- **Description**: A description of the document status.

The Document Status table includes the following predefined records:

Code	Description
ACTIVE	Active
INACTIVE	Inactive

Document Type Table

Records in the Document Type table represent specific types of documents that can be associated with contracts. Records in this table include the following information:

- **Document Type Code**: A unique code that identifies the document type.
- **Description**: A description of the document type.

The Document Type table includes the following predefined records:

Code	Description
SIGNEDPROPOSAL	Signed Proposal
REDLINEDCONTRACT	Red-Lined Contract
LETTEROFCREDIT	Line of Credit
OTHER	Other

Setting Up Other Tables

In addition to contractee and document tables, there are other tables that need to be set up during the configuration of Oracle Utilities Quotations Management. These include the following tables in the Oracle Utilities Data Repository:

- **Contract Category Table**
- **Contract External Status Table**
- **Contract Status Table**

Contract Category Table

Records in the Contract Category table represent specific categories of contracts, such as amendments, subcontracts, offers, etc,. Records in this table include the following information:

- **Contract Category Code**: A unique code that identifies the contract category.
- **Description**: A description of the contract category.

The Contract Category table includes the following predefined records:

Code	Description
CONTRACT	Contract
SUBCONTRACT	Sub-Contract
AMENDMENT	Amendment
OFFER	Offer
BILLING	Billing

Contract External Status Table

Records in the Contract External (Ext.) Status table represent specific external statuses that can be associated to contracts, used for informational purposes. For instance, if a contract has been submitted to a counter party (a customer) and is in the process of being reviewed, a contract analyst could assign an external status of "Customer Review" to the contract. This informs anyone viewing the contract that the contract is currently being reviewed by the customer. Records in this table include the following information:

- **External Contract Status Code**: A unique code that identifies the external contract status.
- **Description**: A description of the external contract status.

The Contract External Status table includes the following predefined records:

Code	Description
INPROCESS	In Process
AWAITINGRISK	Awaiting Risk Approval
READYTOSUBMIT	Ready to Submit
SUBMITTED	Submitted
AWAITINGCREDIT	Awaiting Credit
READYTOEXECUTE	Ready to Execute

Code	Description
EXECUTED	Executed
REJECTED	Rejected
REVISED	Revised
TERMINATED	Terminated
OFFEREXPIRED	Offer Expired

Contract Status Table

Records in the Contract Status table represent specific statuses that can be associated to contracts in the database. Records in this table include the following information:

- **Contract Status Code**: A unique code that identifies the contract status.
- **Description**: A description of the contract status.

The Contract Status table includes the following predefined records:

Code	Description
INPROCESS	In Process
INAPPROVAL	In Approval
APPROVED	Approved
SUBMITTED	Submitted
EXECUTED	Executed
ARCHIVED	Archived
REVISED	Revised
TERMINATED	Terminated
REJECTED	Rejected

Chapter 6

Setting Up Terms, Products, and Contract Types

This chapter describes how to set up terms, products, and contract types used when calculating pricing contracts, including:

- **Setting Up Terms**
- **Setting Up Products**
- **Setting Up Contract Types**

You set up terms and products in the Oracle Utilities Data Repository using the Data Manager application.

You set up Contract Types using the web-enabled Oracle Utilities Quotations Management application. See **Chapter 5**: **Setting Up Contract Terms, Contract Products, and Contract Types** in the *Oracle Utilities Quotations Management User's Guide* for more information.

Setting Up Terms

Terms are specific conditions and/or circumstance required to perform pricing calculations. For example, a pricing contract might require a specific margin, a minimum monthly volume of usage, or a date from which prices are in effect. These types of details and conditions are stored as Terms in the Oracle Utilities Data Repository.

Setting up terms involves creating records in the following tables in the Oracle Utilities Data Repository:

- **Term Category Table**
- **Term Status Table**
- **Term Type Table**
- **Terms Table**

Term Category Table

Records in the Term Category table represent specific categories of terms. Records in this table include the following information:

- **Term Category Code**: A unique code that identifies the document category.
- **Description**: A description of the document category.

The Term Category table includes the following predefined records:

Code	Description
CONTRACT	Contract
PRODUCT	Product

Term Status Table

Records in the Term Status table represent specific status that can be associated to terms in the database. Records in this table include the following information:

- **Term Status Code**: A unique code that identifies the term status.
- **Name**: The name of the term type.

The Term Status table includes the following predefined records:

Code	Description
ACTIVE	Active
PENDING	Pending
INACTIVE	Inactive

Term Type Table

Records in the Term Type table represent specific types of terms that can be used in pricing contract calculations. Records in this table include the following information:

- **Term Type Code**: A unique code that identifies the term type.
- **Name**: The name of the term type.

The Term Type table includes the following predefined records:

Code	Description
BIDTOASK	Bid to Ask
DEPOSITAMOUNT	Deposit Amount
ECOCREDIT	Eco Credit
EFFECTIVEPRICESFROM	Effective Prices From
JOINCREDIT	Join Credit
MARGIN	Margin
MINIMUMMONTHLYVOLUME	Minimum Monthly Volume
NOCREDIT	No Credit
OFFPEAKMARGIN	Off Peak Margin
ONPEAKMARGIN	On Peak Margin
PRICINGPERIOD	Pricing Period
RENEWALOPTION	Renewal Option
TAILLE	Taille
CALCULATEDAMOUNT	Calculated Amount

Terms Table

Records in the Terms table represent individual terms that can be used in pricing contract calculations. Records in this table include the following information:

- **Start Time**: The start time of the term in Oracle Utilities Quotations Management. This is the time after which the term is available for use in pricing calculations.
- **Stop Time**: The stop time of the term in Oracle Utilities Quotations Management. This is the time before which the term is available for use in pricing calculations.
- **Term Type**: The term's type, from the Term Type table.
- **Category**: The category the term falls into. There are two term categories:
  - PRODUCT: Product terms are terms related to a specific product or products. Terms with a category of "PRODUCT" can be associated to a product in Terms screen, accessible from the Items tab of the Contract screen of the Oracle Utilities Quotations Management web application.
  - CONTRACT: Contract terms apply to an entire contract. Terms with a category of "CONTRACT" can be applied to a contract from the Terms tab of the Contract screen of the Oracle Utilities Quotations Management web application.
- **Status**: The status of the term.
- **Default Start Time**: *Optional.* The default start time of the term, if applicable.
- **Default Stop Time**: *Optional.* The default stop time of the term, if applicable.
- **Default Value**: *Optional.* The default value of the term, if applicable.

- **Style View**: An XSL string that defines the user interface when entering values for the term. See **Style View** below for more information. When entering the Style View, it's best to create the XSL string using an XML or text editor, and then paste the string into the field.

The Terms table includes predefined records that correspond to the records in the Term Type table.

Style View

As noted above, the Style View field on the Terms table contains an XSL string that defines the user interface for the term.

Field Types

There are three types of fields that can be included in the Style View

- **Value -Text Field**: This is a simple text entry field, which can include text, numbers, or dates. Values entered in these fields are stored as strings in the Oracle Utilities Data Repository. Dates are stored as values when they are used as references to a date, not as part of calculation. If contract calculations need to use the actual date, use the Date field instead.
- **Value List (drop-down)**: This is a drop-down list comprising one or more values. The specific values available are defined in the XSL string.Values entered in these fields are stored as strings in the Oracle Utilities Data Repository.
- **Date**: This is a date/time field, and includes the same calendar control used on other screens in the Oracle Utilities Energy Information Platform. Values entered in these fields are stored as datetimes in the Oracle Utilities Data Repository.

Example - Margin

The following XSL string could be used to define a term called "Margin" that contains a single value field.

```
<table x:version="1.0" xmlns:x="http://www.w3.org/1999/XSL/Transform"
cellspacing="6" class="FormTable" xmlns:i="urn:ls-i18n-formatter">
  <th>Margin</th>
  <td>
  <input name="X_VAL">
    <x:attribute name="value"><x:value-of select="//@VAL"/></x:attribute>
  </input>
  </td>
</table>
```

Style View Elements

Style View XSL strings use the <table> and related HTML elements, including the following:

- <table>: The root element of the string. This element is closed using the </table> tag.
- <tr>: Defines a row on the user interface (this corresponds to a row in a table). This element is closed using the </tr> tag. This element is only used if the term has more than one value.
- <th>: Defines the heading/name for a specific value on the user interface. This element is closed using the </th> tag.
- <td>: Defines the data for the value on the user interface. This element contains the XSL that defines the data types and attributes for the value. This element is closed using the </td> tag.

Creating Style View XSL Strings

To create a Style View XSL string for a Term, start with the opening and closing <table> elements. Each Style View XSL string should begin with the following element.

```
<table x:version="1.0" xmlns:x="http://www.w3.org/1999/XSL/Transform"
cellspacing="6" class="FormTable" xmlns:i="urn:ls-i18n-formatter">
```

and end with the following element:

```
</table>
```

Next, for each value needed for the term, include the appropriate <th> and <td> elements. Remember to include <tr> elements if the term has more than one value. Examples of each type of value are provided below.

Value -Text Field

Value text fields are simple text entry fields, which can include text, numbers, or dates. Values entered in these fields are stored as strings in the Oracle Utilities Data Repository. XSL used to define a value text field is as follows:

```
<th>[VALUE_NAME]</th>
<td>
  <input name="X_VAL">
    <x:attribute name="value"><x:value-of select="//@VAL"/></x:attribute>
  </input>
</td>
```

where:

- [VALUE_NAME] is the name of the value used by the term (e.g. Margin).

Value List (drop-down)

Value lists are drop-down lists comprising one or more values. The specific values available are defined in the XSL string.Values entered in these fields are stored as strings in the Oracle Utilities Data Repository. XSL used to define a value list field is as follows:

```
<th>[VALUE_NAME]</th>
<td>
  <select name="X_VAL">
    <option value=""></option>
    <option value="[VALUE_1]">
      <x:if test="//@VAL[.='[VALUE_1]']">
        <x:attribute name="selected">true</x:attribute>
      </x:if>
      <x:text>[VALUE_1]</x:text>
    </option>
    <option value="[VALUE_2]">
      <x:if test="//@VAL[.='[VALUE_2]']">
        <x:attribute name="selected">true</x:attribute>
      </x:if>
      <x:text>[VALUE_2]</x:text>
    </option>
  </select>
</td>
```

where:

- [VALUE_NAME] is the name of the value used by the term (e.g. Pricing Period).
- [VALUE_1] is the first value in the drop-down list (e.g. Annual).
- [VALUE_2 is the second value in the drop-down list (e.g. Semi-Annual).

To include additional values in the drop-down list, include additional <option> elements for each.

Date

Dates are date/time fields, and includes the same calendar control used on other screens in the Oracle Utilities Energy Information Platform. Values entered in these fields are stored as datetimes in the Oracle Utilities Data Repository. XSL used to define a date field is as follows:

```
<th>[DATE_NAME]</th>
  <td>
    <input name="X_[DATE_TIME]" class="date">
      <x:attribute name="value"><x:value-of select="i:FD(//
      @[DATE_TIME])"/>/<x:value-of select="i:FD(//@[DATE_TIME])"/>/
      <x:value-of select="i:FD(//@[DATE_TIME])"/></x:attribute>
    </input>
  </td>
```

where:

- [DATE_NAME] is the name of the date used by the term (e.g. Effective Prices From).
- [DATE_TIME] is the type of datetime (e.g. STARTTIME or STOPTIME)

Example - Pricing Period

The example below is an XSL string that defines a term called "Pricing Period", and contains a value list, a Start Time, and a Stop Time.

```
<table x:version="1.0" xmlns:x="http://www.w3.org/1999/XSL/Transform"
cellspacing="6" class="FormTable" xmlns:i="urn:ls-i18n-formatter">
<tr>
  <th>Pricing Period</th>
  <td>
    <select name="X_VAL">
      <option value=""></option>
      <option value="Annual">
        <x:if test="//@VAL[.='Annual']">
          <x:attribute name="selected">true</x:attribute>
        </x:if>
        <x:text>Annual</x:text>
      </option>
      <option value="Semi Annual">
        <x:if test="//@VAL[.='Semi Annual']">
          <x:attribute name="selected">true</x:attribute>
        </x:if>
        <x:text>Semi Annual</x:text>
      </option>
    </select>
  </td>
</tr>
<tr>
  <th>Start Date</th>
  <td>
    <input name="X_STARTTIME" class="date">
      <x:attribute name="value"><x:value-of select="i:FD(//@STARTTIME)"/>/
      <x:value-of select="i:FD(//@STARTTIME)"/>/<x:value-of select="i:FD(//
      @STARTTIME)"/></x:attribute>
    </input>
  </td>
</tr>
<tr>
  <th>Stop Date</th>
  <td>
    <input name="X_STOPTIME" class="date">
```

```
      <x:attribute name="value"><x:value-of select="i:FD(//@STOPTIME)"/>/
      <x:value-of select="i:FD(//@STARTTIME)"/>/<x:value-of select="i:FD(//
      @STARTTIME)"/></x:attribute>
    </input>
  </td>
</tr>
</table>
```

Setting Up Products

Products are pricing structures and rates used in contract and pricing calculations. Each account included in an offer has an associated product. Setting up products involves defining the product using the Oracle Utilities Rules Language, and creating records in the following tables in the Oracle Utilities Data Repository:

- **Product Table**
- **Product Terms Table**

Defining the Product

Products used by Oracle Utilities Quotations Management are defined using the Oracle Utilities Rules Language and related records in the Oracle Utilities Data Repository. These records and the Rules Language used by the product are configured using Data Manager., including:

- **Rate Forms**: Records in the Rate Form and Rate Form Version tables. Products must be define in **Riders**. The Operating Company and Jurisdiction for this rider MUST both be LODESTAR. Oracle Utilities recommends using the following naming conventions for contract rate forms:

  LSPE_<PRODUCT_NAME>_RDR

  where <PRODUCT_NAME> is the ID of the product in the Product Table (see **Defining the Product** on page 6-8).

- **Rules Language**: The actual Rules Language configuration that defines the product's pricing algorithms. Refer to the *Oracle Utilities Rules Language User's Guide* and *Oracle Utilities Rules Language Reference Guide* for more information.
- **Lists**: Table-column lists used by the Rules Language calculations.
- **Factors**: Factors used by the Rules Language calculations, stored in the Factor and Factor Value tables. Refer to the *Oracle Utilities Rules Language User's Guide* for more information.
- **Overrides**: Overrides (or special events) used by the Rules Language calculations, stored in the Override and Override Value tables. Refer to the *Oracle Utilities Rules Language User's Guide* for more information.

Product Table

Records in the Product table represent individual products that can be associated to accounts for a contract and used in contract calculations. Records in this table include the following information:

- **ID**: A unique id for the product.
- **Description**: A description of the product.
- **Start Time**: The start time of the product in Oracle Utilities Quotations Management. This is the time after which the product is available for use in pricing calculations.
- **Stop Time**: The stop time of the product in Oracle Utilities Quotations Management. This is the time before which the product is available for use in pricing calculations.
- **Rate Form**: The rate form that defines the product, from the Rate Form table.

Product Terms Table

Records in the Product Terms table link terms (with a Category of "PRODUCT") to products in the Oracle Utilities Data Repository. This relationship determines which terms are available for a particular product on the Product Terms screen, accessible from the Items tab of the Contract screen of the Oracle Utilities Quotations Management web application. Records in this table include the following information:

- **Product**: The product, from Product table.
- **Term**: The term, from the Terms table.
- **Start Time**: The start time of the product term in Oracle Utilities Quotations Management. This is the time after which the product term is available for use in pricing calculations.
- **Stop Time**: The stop time of the product term in Oracle Utilities Quotations Management. This is the time before which the product term is available for use in pricing calculations.
- **Term Type**: The term type for the product term, from the Term Type table. **Note**: This must match the Term Type of the Term.
- **Is Standard?**: A flag that indicates if the product term is the "standard" product term for the product.
- **Is Required?**: A flag that indicates if the product term is required for the product.
- **Default Start Time**: *Optional.* The default start time of the product term, if applicable.
- **Default Stop Time**: *Optional.* The default stop time of the product term, if applicable.
- **Default Value**: *Optional.* The default value of the product term, if applicable.

Setting Up Contract Types

Contract types are contract templates used to create contracts using the Oracle Utilities Billing Component - Contract Management application. Setting up contract types allows you create default settings for different types of contracts, and involves creating the following data in the Oracle Utilities Data Repository

- **Contract Type Records**
- **Contract Type Documents**
- **Contract Type Terms**
- **Contract Type Approvals**

Contract Type Records

Contract type records represent individual contract types and are stored in the Contract Type table. Records in this table contain the following information:

- **Contract Type Code**: A unique code that identifies the contract type.
- **Contractee**: individuals within a business who can create contracts to be signed with counter parties (parties outside the business). See **Setting Up Contractee Tables** on page 5-2 for more information about contractees.
- **Description**: a description of the contract type.
- **Contract Category**: the specific category (contract, amendment, subcontract, offer, etc.) appropriate for the contract type. See **Setting Up Other Tables on page 5-6** for more information about contract categories.
- **Start Time/Stop Time**: The start/stop time for the contract type in Oracle Utilities Billing Component - Contract Management.
- **Work Queue Type**: the work queue type used to implement the approval process, (only used with contract types that require an approval process). See **Contract Type Approvals** for more information.
- **Rate Form**: an optional rate form (of type CONTRACT) associated with the contract type that contains Contract Type-specific business logic. See **Adding Contract Type-Specific Business Logic** on page 7-3 for more information.

Contract Types are created using the **Add a Contract Type** option of the Oracle Utilities Billing Component - Contract Management application, and are maintained on the **Basics** tab on the Contract Type screen.

Contract Type Documents

Contract type document records represent relationships between contract types and document types, and are stored in the Contract Type Document Relationship table. Records in this table contain the following information:

- **Contract Type**: the contract type associated with the document type, from the Contract Type table.
- **Document Type**: the document type associated with the contract type, from the Document Type table. See **Setting Up Document Tables** on page 5-4 for more information about document types.
- **Document Category**: the specific category (contracts, proposals, etc.) appropriate for the document type, from the Document Category table. See **Setting Up Document Tables** on page 5-4 for more information about document categories.
- **Standard**: a flag that indicates if the document type is standard for contracts of this contract type.
- **Start Time**/**Stop Time**: The start/stop time for the contract type/document type relationship in Oracle Utilities Billing Component - Contract Management.
- **Required**: a flag that indicates if the document type is required for contracts of this contract type.

Contract Type Document records are created on the **Documents** tab on the Contract Type screen.

Contract Type Terms

Contract type term records represent relationships between contract types and terms, and are stored in the Contract Type Term table. Records in this table contain the following information:

- **Contract Type Code**: the contract type associated with the term, from the Contract Type table.
- **Term**: the term associated with the contract type, from the Terms table. See **Setting Up Terms** on page 6-2 for more information about terms.
- **Start Time**/**Stop Time**: The start/stop time for the contract type/term relationship in Oracle Utilities Billing Component - Contract Management.
- **Standard**: a flag that indicates if the term is standard for contracts of this contract type.
- **Required**: a flag that indicates if the term is required for contracts of this contract type.
- **Calculated**: a flag that indicates if the term is calculated.
- **Default Value**: the default value for the term when used with this contract type.
- **Default Start Time**/**Default Stop Time**: The default start/default stop time for the contract type/term relationship in Oracle Utilities Billing Component - Contract Management.

Contract Type Term records are created on the **Terms** tab on the Contract Type screen.

Contract Type Approvals

Contract Type Approvals are approval procedures associated with contract types. For instance, contracts created from a specific contract type might required approval by certain individuals before being submitted to a counter party. Contract type approvals are configured using the Work Queues functions available via the Oracle Utilities Energy Information Platform.

Contract types can have an associated work queue type that defines the approval process for contracts of that type. Contract Type Approvals are viewed on the **Approvals** tab on the Contract screen.

Creating Contract Type Approvals

To create a contract type approval, first create a work queue type that includes the approval process you wish to associate to the contract type.

Contract type approvals (work queue types) can be configured to update the status of the contract upon approval (Approved) or rejection (Rejected). This feature can be enabled by entering "LSCM.WorkQueueCloseCB" in the Close Callback ProgID column in the Work Queue Type table.

See **Setting Up Work Queue Types** on page 5-12 of the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about setting up approval procedures and work queue types.

How Approvals Work

When a user selects the **Ready for Approval** action from the **Actions** drop-down list on the Contract screen, a work queue item (of the associated work queue type) and a corresponding approval procedure is generated. As the contract is reviewed by the appropriate individual(s), the approvals related to the contract are either Approved or Rejected.

Chapter 7

Adding Customized Business Logic to Oracle Utilities Quotations Management

This chapter describes how to add customized business logic to Oracle Utilities Quotations Management, including:

- **Customizing Oracle Utilities Quotations Management Processes**
- **Oracle Utilities Rules Language - Term Functions**
- **Customizing Oracle Utilities Quotations Management Calculations and Contracts**
- **Customizing the Contracts Screen**

Customizing Oracle Utilities Quotations Management Processes

This section describes how to customize the pricing contract processes run by Oracle Utilities Quotations Management, including:

- **Contract Processes**
- **Pricing Processes**

Contract Processes

Each of the contract processes performed by Oracle Utilities Quotations Management (creating a contract, calculating terms, revising a contract, creating contract approvals, submitting a contract, executing a contract, and terminating a contract) is performed by pre-configured Rules Language rate schedules and riders (described in **Chapter 4**: **Oracle Utilities Quotations Management Rules Language Schedules, Riders, and Lists**).

In addition to performing pre-configured application logic, these rate schedules and riders allow users to include customized business logic at any point in the contracting process, including:

- Creating a Contract (LSCM_CREATE_CONTRACT)
- Calculating Terms (LSCM_CALCULATE_TERMS)
- Revising a Contract (LSCM_REVISE_CONTRACT)
- Creating Contract Approvals (LSCM_CONTRACT_APPROVAL)
- Submitting a Contract (LSCM_SUBMIT_CONTRACT)
- Executing a Contract (LSCM_EXECUTE_CONTRACT)
- Terminating a Contract. (LSCM_TERMINATE_CONTRACT)

The specific Rules Language rate schedule that performs each of these processes is noted in parentheses.

Several of these rate schedules include riders (supplied by Oracle Utilities) that can be used to add customized logic to that particular process using the Rules Language. The names of these riders are as follows:

- Calculate Custom Terms (LSCM_CALCULATE_TERMS_RDR)
- Submit Contract (LSCM_SUBMIT_CONTRACT_RDR)
- Execute Contract (LSCM_EXECUTE_CONTRACT_RDR)
- Terminate Contract. (LSCM_TERMINATE_CONTRACT_RDR)

The other processes (Creating a Contract, Revising a Contract, and Creating Contract Approvals) do not include supplied riders. When adding custom logic to these processes, you must create a rider that contains the customized Rules Language logic and include that rider in the appropriate rate schedule.

Adding Custom Logic to Contract Processes

To add customized business logic to one or more of the above contract processes, create a new version of the appropriate supplied rider, and add appropriate Rules Language configuration. See **How to open a new rate form version:** on page 2-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form versions.

For example, to add customized logic to the Submit Contract process, you would create a new version of the LSCM_SUBMIT_CONTRACT_RDR and include the custom Rules Language configuration to perform the desired logic.

Adding Custom Logic without Supplied Riders

To add customized business logic to Creating a Contract, Revising a Contract, or Creating Contract Approvals processes, use the following procedure:

1. Create a rate form record of type RIDER for each process you wish to customize. See **How to create a rate form record:** on page A-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form records. The Operating Company and Jurisdiction for this rider MUST both be LODESTAR. Oracle Utilities recommends using the following naming conventions for the riders for these processes:
   - Creating a Contract (LSCM_CREATE_CONTRACT_RDR)
   - Revising a Contract (LSCM_REVISE_CONTRACT_RDR)
   - Creating Contract Approvals (LSCM_CONTRACT_APPROVAL_RDR)
2. Create a new version of the rider, and add appropriate Rules Language configuration. See **How to open a new rate form version:** on page 2-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form versions.
3. Include the rider in the appropriate place in the appropriate rate schedule using the INCLUDE Rules Language statement. See **Include Statement** on page 3-23 in the *Oracle Utilities Rules Language Reference Guide* for more information about using the INCLUDE statement.

For example, to add customized logic to the Create Contract process, you would first create a new record in the Rate Form table for the rider (LSCM_CREATE_CONTRACT_RDR). Next, you would create a new version of the LSCM_CREATE_CONTRACT_RDR and insert custom Rules Language configuration to perform the desired logic. Lastly, you would use the INCLUDE statement to include the rider at the appropriate place in the LSCM_CREATE_CONTRACT the rate schedule.

Notes:

- When creating the new rider version of a supplied rider, be sure that the date for the version is later than the date of the supplied rider.
- Values for any identifiers used in the rider must be set either in the parent rate schedule or in the rider itself.

Adding Contract Type-Specific Business Logic

You can also create custom business logic that applies to specific processes for all contracts of a certain Contract Type. For example, you might want to perform specific business logic when submitting all contracts of a certain type. Use the following procedure to add Contract Type-specific business logic:

1. Create a rate form record of type RIDER for each Contract Type you wish to customize. See **How to create a rate form record:** on page A-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form records. The Operating Company and Jurisdiction for this rider MUST both be LODESTAR. Oracle Utilities recommends using the following naming conventions for contract rate forms:
   - LSCM_<CONTRACT_TYPE_CODE>_RDR

     where <CONTRACT_TYPE_CODE> is the Contract Type Code for the Contract Type.
2. Create a new version of the rider. See **How to open a new rate form version:** on page 2-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form versions.
3. Create the following SELECT statement in the rider:

```
SELECT RATE_SCHEDULE_CODE
```

4. Within this SELECT statement, create WHEN clauses that correspond to each of the contract processes that will have contract type-specific business logic. These clauses should list the name of the rate schedule that corresponds to the contract processes that will have contract type-specific business logic. (WHEN "LSCM_SUBMIT_CONTRACT", etc.). See **Select Expression Statement** on page 3-31 in the *Oracle Utilities Rules Language Reference Guide* for more information about using the SELECT statement.
5. Add Rules Language configuration for the appropriate business logic in the corresponding WHEN clause of the rider. For instance, include logic related to submitting a contract in the WHEN "LSCM_SUBMIT_CONTRACT" clause of the rider.
6. Associate the contract rate form with the appropriate Contract Type on the Basics tab of the Contract Type screen of the web-enabled Oracle Utilities Quotations Management application. See **Basics Tab** on page 6-6 of the *Oracle Utilities Quotations Management User's Guide* for more information.
7. Create a list query that retrieves the Contract Type associated with the contract (LSCM_GET_CONTRACT_TYPE), and use the LISTVALUE function to retrieve the Rate Form (RATEFORMCODE) associated with the contract type, and assign that value to an identifier (CONTRACT_TYPE_RIDER).
8. Use the CALL statement to call the CONTRACT_TYPE_RIDER in the corresponding rider (i.e. LSCM_SUBMIT_CONTRACT_RDR) See **Call Statement** on page 3-3 in the *Oracle Utilities Rules Language Reference Guide* for more information about using the CALL statement.

For example, to add customized logic to the Submit Contract process for all contracts of type "ELECTRIC", you would perform the following steps:

1. Create a new record in the Rate Form table for the contract (LSCM_ELECTRIC_RDR).
2. Create a new version of the LSCM_ELECTRIC_RDR, and include the following SELECT statement in the rider.

   ```
   SELECT RATE_SCHEDULE_CODE
   WHEN "LSCM_SUBMIT_CONTRACT"
   END SELECT;
   ```

3. Include custom Rules Language configuration to perform the desired logic within the WHEN "LSCM_SUBMIT_CONTRACT" clause of the LSCM_ELECTRIC_RDR rider.
4. Associate the LSCM_ELECTRIC_RDR contract with the ELECTRIC Contract Type on the Basics tab of the Contract Type screen.
5. Include the following statements at the appropriate place in the LSCM_SUBMIT_CONTRACT_RDR rider.

   ```
   CT_TYPE = LISTVALUE ("LSCM_GET_CONTRACT_TYPE");
   CONTRACT_TYPE_RIDER = CT_TYPE.RATEFORMCODE;
   CALL CONTRACT_TYPE_RIDER;
   ```

   These statements retrieve the rate form associated with the contract type (LSCM_ELECTRIC_RDR) and calls the rider. The SELECT statement then dictates the portion of the rider used in processing (in this case, the portion in the "LSCM_SUBMIT_CONTRACT" clause).

Pricing Processes

In addition to the contract processes described above, Oracle Utilities Quotations Management also performs a number of other processes via pre-configured Rules Language rate schedules and riders (described in **Chapter 4**: **Oracle Utilities Quotations Management Rules Language Schedules, Riders, and Lists**). These processes include:

- Validate Usage Data (Standard) (LSPE_VAL_USAGE)
- Validate Usage Data (Loose) (LSPE_VAL_USAGE_LOOSE)
- Estimate and Normalize Usage Data (LSPE_EST_USAGE)
- Validate Forecast (LSPE_VAL_FORECAST)
- Finalize Forecast (LSPE_FINALIZE_FORECAST)
- Calculate Pricing (LSCM_CALCULATE_PRICE)
- Apply Product Terms (LSCM_SHARE_ITEM_TERMS)

All of these rate schedules include riders (supplied by Oracle Utilities) that can be used to add customized logic to that particular process using the Rules Language. The names of these riders are as follows:

- Validate Usage (Standard) (LSPE_VAL_USAGE_RDR)
- Validate Usage (Loose) (LSPE_VAL_USAGE_LOOSE_RDR)
- Estimate Usage (LSPE_EST_USAGE_RDR)
- Normalize Usage (LSPE_NORM_USAGE_RDR)
- Validate Forecast (LSPE_VAL_FORECAST_RDR)
- Finalize Forecast (LSPE_FINALIZE_FORECAST_RDR)
- Calculating Pricing (LSCM_CALCULATE_PRICE_RDR)
- Load Attributes (LSCM_LOAD_ATTRIBUTES_RDR)
- Load Item Attributes (LSCM_LOAD_ITEM_ATTRIBUTES_RDR)
- Apply Product Terms (LSCM_SHARE_ITEM_TERMS_RDR)

Adding Custom Logic to Pricing Processes

To add customized business logic to one or more of the above Oracle Utilities Quotations Management processes, create a new version of the appropriate supplied rider, and add appropriate Rules Language configuration. See **How to open a new rate form version:** on page 2-2 in the *Oracle Utilities Rules Language User's Guide* for more information about creating new rate form versions.

For example, to add customized logic to the Validate Usage (Standard) process, you would create a new version of the LSPE_VAL_USAGE_RDR and include the custom Rules Language configuration to perform the desired logic.

Adding Item Attributes to Pricing Contracts

When creating pricing contracts, you can also define a number of attributes for each contract item (account) in the pricing contract, including the following:

- Cost of Good Sold (COGS)
- Margin
- Rate
- Max Quantity
- Quantity

- Probability
- Offer Valid Time

The riders supplied by Oracle Utilities include default values for these attributes. Values can be assigned to these attributes (based on either specified values or calculations) through configuration of the LSCM_LOAD_ATTRIBUTES rider (for contract-level attributes) and the LSCM_LOAD_ITEM_ATTRIBUTES rider (for contract item-level attributes).

Custom attributes beyond those listed above can be added to pricing contracts and contract items by adding columns to the Contract and Contract Item tables via meta-data modifications to the Oracle Utilities Data Repository, and assigning values to the attributes in the LSCM_LOAD_ATTRIBUTES and LSCM_LOAD_ITEM_ATTRIBUTES riders (respectively).

Adding Product-specific Calculations to Pricing Contracts

Pricing contracts will often require product-specific calculations. This allows pricing contract calculations to be based on the specific products associated to each account in the contract.Use the following procedure to add product-specific calculations:

1. Define the product as described under **Setting Up Products** in **Chapter 6**: **Setting Up Terms, Products, and Contract Types**.
2. Create a list query that retrieves the Product associated with a supplied contract item (LSPE_GET_PRODUCT), and use the LISTVALUE function in the LSCM_CALCULATE_PRICE_RDR to retrieve the Rate Form (RATEFORMCODE) associated with the product, and assign that value to an identifier (ACCOUNT_PRODUCT_RIDER).
3. Use the CALL statement to call the ACCOUNT_PRODUCT_RIDER in the LSCM_CALCULATE_PRICE_RDR rider See **Call Statement** on page 3-3 in the *Oracle Utilities Rules Language Reference Guide* for more information about using the CALL statement.

For example, to include calculations for a product called "FLAT_CHARGE", you would perform the following steps:

1. Create the rider and associated data (Factors, Overrides, and Lists) for the product (LSPE_FLAT_CHARGE_RDR). This rider should include the logic required for pricing calculations associated with the product.
2. Include the following statements at the appropriate place in the LSCM_CALCULATE_PRICE_RDR rider.

   ```
   PRODUCT = LISTVALUE ("LSPE_GET_PRODUCT");
   ACCOUNT_PRODUCT_RIDER = PRODUCT.RATEFORMCODE;
   CALL ACCOUNT_PRODUCT_RIDER;
   ```

   These statements retrieve the rate form associated with the product (LSPE_FLAT_CHARGE_RDR) and calls the rider.

Oracle Utilities Rules Language - Term Functions

Term functions are used to retrieve and save terms and term details to and from the Oracle Utilities Data Repository.

Term Function Tail Identifiers

Term functions use the following tail identiers for database columns:

Tail Identifier	Table	Field
CONTRACTID	LSCMCONTRACT	CONTRACTID
REVISION	LSCMCONTRACT	REVISION
ACCOUNTID	ACCOUNT	ACCOUNTID
SERVICEPOINT	LSSERVICEPOINT	SERVICEPOINTID
MARKETID	LSMARKET	MARKETID
SERVICETYPE	LSSERIVICETYPE	SERVICETYPE
PRODUCTID	LSCMPRODUCT	PRODUCTID
PRODUCTSTART	LSCMPRODUCT	STARTTIME
PRODUCTSTOP	LSCMPRODUCT	STOPTIME
GROUPID	LSCMCONITEMGROUP	GROUPID
TERMTYPE	LSTERMTYPE	TERMTYPECODE
TERMCATEGORY	LSTERMCATEGORY	TERMCATEGORYCODE
TERMSTART	LSTERM	STARTTIME
TERMSTOP	LSTERM	STOPTIME
STARTTIME	Term Table*	STARTTIME
STOPTIME	Term Table*	STOPTIME
VAL	Term Table*	VAL
VALNUM	Term Table*	VALNUM
VALDATE	Term Table*	VALDATE
ISSTANDARD	LSCMCONTRACTTERM	ISSTANDARD
ISREQUIRED	LSCMCONTRACTTERM	ISREQUIRED
ISCALCULATED	LSCMCONTRACTTERM	ISCALCULATED
PERIOD	LSCMCONITEMDTLS	PERIOD

*Any of the following tables used to store terms:

- LSCMCONTRACTTERM (Contract Terms)
- LSCMCONITEMTERM, LSCMCONITEMPRDTERM, LSCMCONITEMDTLS (Contract Item Terms)

- LSCMCONITEMGROUPPRDTERM (Contract Group Terms)

LOADCONTRACTTERM Function

Purpose

The LOADCONTRACTTERM function loads a single contract term from the Contract Terms table in the Oracle Utilities Data Repository. The function returns a stem identifier containing the retrieved term.

Format

```
<output_stem> = LOADCONTRACTTERM(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to retrieve:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved
  - TERMTYPE: The term type for the term to be retrieved
  - TERMCATEGORY: The term category for the term to be retrieved
  - STARTTIME: The start time of the term to be retrieved

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <output_stem> is a stem identifier with following tail identifiers representing columns from the Contract Term table:
  - CONTRACTID: The contract ID of the contract for the retrieved term
  - REVISION: The revision number of the contract for the retrieved term
  - TERMSTART: The start time of the term
  - TERMSTOP: The stop time of the term
  - TERMTYPE: The term type for the retrieved term
  - TERMCATEGORY: The term category for the retrieved term
  - STARTTIME: The start time of the retrieved term
  - STOPTIME: The stop time of the retrieved term
  - VAL: The text value of the retrieved term
  - VALNUM: The numeric value of the retrieved term
  - VALDATE: The date value of the retrieved term
  - ISSTANDARD: The Is Standard flag of the retrieved term
  - ISREQUIRED: The Is Required flag of the retrieved term
  - ISCALCULATED: The Is Calculated flag of the retrieved term
  - Any custom columns on the Contract Term table. The tail identifier used will be the same as the custom column name.
  - ERRORRETURN: Return code of the function call. An error occurs if no term record is found, if multiple terms records are found, or if a custom column on the Contract Term

table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7). Return codes are as follows:

- 0 - Success
- 1 - No Record Found
- 2 - Multiple Records Found
- 3 - Column name conflict with pre-defined tail

Note: Any errors returned will also appear on the output report.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve the MARGIN,CONTRACT term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1.

```
LOAD_TERM.CONTRACTID = "Customer_Pricing_01";
LOAD_TERM.REVISION = "1";
LOAD_TERM.TERMTYPE = "MARGIN";
LOAD_TERM.TERMCATEGORY = "CONTRACT";
LOAD_TERM.STARTTIME = "01/01/2008 00:00:00";
CONTRACT_TERM_DTLS = LOADCONTRACTTERM(LOAD_TERM);
```

This function would return the following tail identifiers for the "CONTRACT_TERM_DTLS" stem identifier:

Tail Identifiers	Value
CONTRACTID	"Customer_Pricing_01"
REVISION	"1"
TERMSTART	"01/01/2006 00:00:00"
TERMSTOP	NULL
TERMTYPE	"MARGIN"
TERMCATEGORY	"CONTRACT"
STARTTIME	"01/01/2006 00:00:00"
STOPTIME	NULL
VAL	NULL
VALNUM	"5"
VALDATE	NULL
ISSTANDARD	"Yes"
ISREQUIRED	"Yes"
ISCALCULATED	"No"

LOADCONTRACTTERMALL Function

Purpose

The LOADCONTRACTTERMALL function loads all contract terms for a specified contract from the Contract Terms table in the Oracle Utilities Data Repository. This function creates one or more stem identifiers containing the retrieved terms. The function returns zero if successful, and returns an integer (1, 2, 3, or 4) if an error occurs.

Format

```
<error_code> = LOADCONTRACTTERMALL(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the contract from which to retrieve the terms:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <error_code> is the return code of the function call. An error occurs if no term record is found, if multiple terms records are found, if a custom column on the Contract Term table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7), or if the 64-character Rules Language identifier length is exceeded. Return codes are as follows:
  - 0 - Success
  - 1 - No Record Found
  - 2 - Multiple Records Found
  - 3 - Column name conflict with pre-defined tail
  - 4 - Identifier 64-character limit exceeded

  Note: Any errors returned will also appear on the output report.

Stem Identifiers: This function creates one or more stem identifiers that contain the retrieved terms. The stem identifiers are created by concatenating the table name prefix ("CONT"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

```
CONT_MARGIN_CONTRACT
```

If the Category Code is null, then the stem identifier will be created by concatenating the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

```
CONT_MARGIN
```

Stem identifiers are made into array identifiers if there are multiple values of STARTTIME (in the term table) for the same term for the specified contract.

Tail Identifiers: Each stem identifier has the following tail identifiers:

- TERMSTART: The start time of the term
- TERMSTOP: The stop time of the term
- STARTTIME: The start time of the retrieved term

- STOPTIME: The stop time of the retrieved term
- VAL: The text value of the retrieved term
- VALNUM: The numeric value of the retrieved term
- VALDATE: The date value of the retrieved term
- ISSTANDARD: The Is Standard flag of the retrieved term
- ISREQUIRED: The Is Required flag of the retrieved term
- ISCALCULATED: The Is Calculated flag of the retrieved term
- Any custom columns on the Contract Term table. The tail identifier used will be the same as the custom column name.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve all terms for contact "Customer_Pricing_01", revision 1.

```
LOAD_ALL_TERMS.CONTRACTID = "Customer_Pricing_01";
LOAD_ALL_TERMS.REVISION = "1";
ALL_CONTRACT_TERM = LOADCONTRACTTERMALL(LOAD_ALL_TERMS);
```

This function would return a number of stem identifiers, one for each combination of Term Type and Category. For example:

```
CONT_MARGIN_CONTRACT
CONT_DEPOSITAMOUNT_CONTRACT
CONT_EFFECTIVEPRICESFROM_CONTRACT
CONT_ONPEAKMARGIN_PRODUCT
CONT_OFFPEAKMARGIN_PRODUCT
CONT_CUST_CHARGE_TYPE_PRODUCT
...
```

Each of these stem identifiers would contain the above listed tail identifiers.

If there were multiple Start Time values for the same term, these stem identifiers would be array identifiers, each with an upper bound equal to the number of term records returned.

LOADGROUPTERM Function

Purpose

The LOADGROUPTERM function loads a single contract group term from the Contract Item Group Terms table in the Oracle Utilities Data Repository. The function returns a stem identifier containing the retrieved term.

Format

```
<output_stem> = LOADGROUPTERM(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to retrieve:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved
  - TERMTYPE: The term type for the term to be retrieved
  - TERMCATEGORY: The term category for the term to be retrieved
  - STARTTIME: The start time of the term to be retrieved
  - PRODUCTID: The Product ID for the contract item group
  - PRODUCTSTART: The Product start time for the contract item group
  - PRODUCTSTOP: The Product stop time for the contract item group
  - GROUPID: The Group ID for the contract item group

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <output_stem> is a stem identifier with following tail identifiers representing columns from the Contract Item Group Product Term table:
  - CONTRACTID: The contract ID of the contract for the retrieved term
  - REVISION: The revision number of the contract for the retrieved term
  - TERMSTART: The start time of the term
  - TERMSTOP: The stop time of the term
  - TERMTYPE: The term type for the retrieved term
  - TERMCATEGORY: The term category for the retrieved term
  - PRODUCTID: The Product ID for the contract item group
  - PRODUCTSTART: The Product start time for the contract item group
  - PRODUCTSTOP: The Product stop time for the contract item group
  - GROUPID: The Group ID for the contract item group
  - STARTTIME: The start time of the retrieved term
  - STOPTIME: The stop time of the retrieved term
  - VAL: The text value of the retrieved term
  - VALNUM: The numeric value of the retrieved term
  - VALDATE: The date value of the retrieved term

- Any custom columns on the Contract Item Group Product Term table. The tail identifier used will be the same as the custom column name.
- ERRORRETURN: Return code of the function call. An error occurs if no term record is found, if multiple terms records are found, or if a custom column on the Contract Term table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7). Return codes are as follows:
  - 0 - Success
  - 1 - No Record Found
  - 2 - Multiple Records Found
  - 3 - Column name conflict with pre-defined tail

  Note: Any errors returned will also appear on the output report.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve the DEPOSITAMOUNT,CONTRACT group term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1, for group "GROUP_01" and Product "GENERAL_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59).

```
LOAD_GROUP_TERM.CONTRACTID = "Customer_Pricing_01";
LOAD_GROUP_TERM.REVISION = "1";
LOAD_GROUP_TERM.TERMTYPE = "DEPOSITAMOUNT";
LOAD_GROUP_TERM.TERMCATEGORY = "CONTRACT";
LOAD_GROUP_TERM.STARTTIME = "01/01/2008 00:00:00";
LOAD_GROUP_TERM.PRODUCTID = "GENERAL_SERVICE";
LOAD_GROUP_TERM.PRODUCTSTART = "01/01/2007 00:00:00";
LOAD_GROUP_TERM.PRODUCTSTOP = "12/31/2010 23:59:59";
LOAD_GROUP_TERM.GROUPIP = "GROUP_01";
GROUP_TERM_DTLS = LOADCONTRACTTERM(LOAD_GROUP_TERM);
```

This function would return the following tail identifiers for the "GROUP_TERM_DTLS" stem identifier:

Tail Identifiers	Value
CONTRACTID	"Customer_Pricing_01"
REVISION	"1"
TERMSTART	"01/01/2006 00:00:00"
TERMSTOP	NULL
TERMTYPE	"DEPOSITAMOUNT"
TERMCATEGORY	"CONTRACT"
PRODUCTID	"GENERAL_SERVICE"
PRODUCTSTART	"01/01/2007 00:00:00"
PRODUCTSTOP	"12/31/2010 23:59:59"
GROUPID	"GROUP_01"

Tail Identifiers	Value
STARTTIME	"01/01/2006 00:00:00"
STOPTIME	NULL
VAL	NULL
VALNUM	"100"
VALDATE	NULL

LOADGROUPTERMALL Function

Purpose

The LOADGROUPTERMALL function loads all contract group terms for a specified contract and group from the Contract Item Group Terms table in the Oracle Utilities Data Repository. This function creates one or more stem identifiers containing the retrieved terms. The function returns zero if successful, and returns an integer (1, 2, 3, or 4) if an error occurs.

Format

```
<error_code> = LOADGROUPTERMALL(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the contract from which to retrieve the terms:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved
  - PRODUCTID: The Product ID for the contract item group
  - PRODUCTSTART: The Product start time for the contract item group
  - PRODUCTSTOP: The Product stop time for the contract item group
  - GROUPID: The Group ID for the contract item group

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <error_code> is the return code of the function call. An error occurs if no term record is found, if multiple terms records are found, if a custom column on the Contract Term table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7), or if the 64-character Rules Language identifier length is exceeded. Return codes are as follows:
  - 0 - Success
  - 1 - No Record Found
  - 2 - Multiple Records Found
  - 3 - Column name conflict with pre-defined tail
  - 4 - Identifier 64-character limit exceeded

  Note: Any errors returned will also appear on the output report.

Stem Identifiers: This function creates one or more stem identifiers that contain the retrieved terms. The stem identifiers are created by concatenating the table name prefix ("GRUP"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

```
GRUP_MARGIN_CONTRACT
```

If the Category Code is null, then the stem identifier will be created by concatenating the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

```
GRUP_MARGIN
```

Stem identifiers are made into array identifiers if there are multiple values of STARTTIME (in the term table) for the same term for the specified contract.

Tail Identifiers: Each stem identifier has the following tail identifiers:

- TERMSTART: The start time of the term
- TERMSTOP: The stop time of the term
- STARTTIME: The start time of the retrieved term
- STOPTIME: The stop time of the retrieved term
- VAL: The text value of the retrieved term
- VALNUM: The numeric value of the retrieved term
- VALDATE: The date value of the retrieved term
- Any custom columns on the Contract Term table. The tail identifier used will be the same as the custom column name.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve all group terms for contact "Customer_Pricing_01", revision 1 for group "GROUP_01" and Product "GENERAL_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59).

```
LOAD_ALL_GROUP_TERMS.CONTRACTID = "Customer_Pricing_01";
LOAD_ALL_GROUP_TERMS.REVISION = "1";
LOAD_ALL_GROUP_TERMS.PRODUCTID = "GENERAL_SERVICE";
LOAD_ALL_GROUP_TERMS.PRODUCTSTART = "01/01/2007 00:00:00";
LOAD_ALL_GROUP_TERMS.PRODUCTSTOP = "12/31/2010 23:59:59";
LOAD_ALL_GROUP_TERMS.GROUPIP = "GROUP_01";
ALL_CONTRACT_GROUP_TERM = LOADGROUPTERMALL(LOAD_ALL_GROUP_TERMS);
```

This function would return a number of stem identifiers, one for each combination of Term Type and Category. For example:

```
GRUP_MARGIN_CONTRACT
GRUP_DEPOSITAMOUNT_CONTRACT
GRUP_EFFECTIVEPRICESFROM_CONTRACT
GRUP_ONPEAKMARGIN_PRODUCT
GRUP_OFFPEAKMARGIN_PRODUCT
GRUP_CUST_CHARGE_TYPE_PRODUCT
...
```

Each of these stem identifiers would contain the above listed tail identifiers.

If there were multiple Start Time values for the same term, these stem identifiers would be array identifiers, each with an upper bound equal to the number of term records returned.

LOADITEMTERM Function

Purpose

The LOADITEMTERM function loads a single contract item term from the Oracle Utilities Data Repository. This function can retrieve terms from the Contract Item Term table, the Contract Item Product Term table, or the Contract Item Details table. The function returns a stem identifier containing the retrieved term.

Format

```
<output_stem> = LOADITEMTERM(<input_stem>, <table>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to retrieve:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved
  - TERMTYPE: The term type for the term to be retrieved
  - TERMCATEGORY: The term category for the term to be retrieved
  - STARTTIME: The start time of the term to be retrieved
  - ACCOUNTID: The Account ID for the contract item (if applicable)
  - PRODUCTID: The Product ID for the contract item (if applicable)
  - PRODUCTSTART: The Product start time for the contract item (if applicable)
  - PRODUCTSTOP: The Product stop time for the contract item (if applicable)
  - SERVICEPOINT: The Service Point ID for the contract item (if applicable)
  - MARKETID: The Market ID related to the Service Point ID for the contract item (if applicable)
  - SERVICETYPE: The service type related to the Service Point ID for the contract item (if applicable)
  - STOPTIME: The stop time of the term to be retrieved. This tail only applies when retrieving term details from Contract Item Details table.

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <table> is a string that specifies the table from which the term details are to be retrieved:
  - "ITEM": retrieve the term details from the Contract Item Term table
  - "PRODUCT": retrieve the term details from the Contract Item Product Term table
  - "DETAILS" retrieve the term details from the Contract Item Details table
- <output_stem> is a stem identifier with following tail identifiers representing columns from the specified table:
  - CONTRACTID: The contract ID of the contract for the retrieved term
  - REVISION: The revision number of the contract for the retrieved term
  - TERMSTART: The start time of the term
  - TERMSTOP: The stop time of the term
  - TERMTYPE: The term type for the retrieved term

- TERMCATEGORY: The term category for the retrieved term
- ACCOUNTID: The Account ID for the contract item
- PRODUCTID: The Product ID for the contract item
- PRODUCTSTART: The Product start time for the contract item
- PRODUCTSTOP: The Product stop time for the contract item
- SERVICEPOINT: The Service Point ID for the contract item
- MARKETID: The Market ID related to the Service Point ID for the contract item
- SERVICETYPE: The service type related to the Service Point ID for the contract item
- STARTTIME: The start time of the retrieved term
- STOPTIME: The stop time of the retrieved term
- VAL: The text value of the retrieved term
- VALNUM: The numeric value of the retrieved term
- VALDATE: The date value of the retrieved term
- PERIOD: The period for the retrieved term. This tail is only returned when retrieving terms from the Contract Item Details table.
- Any custom columns on the specified table. The tail identifier used will be the same as the custom column name.
- ERRORRETURN: Return code of the function call. An error occurs if no term record is found, if multiple terms records are found, or if a custom column on the specified table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7). Return codes are as follows:
  - 0 - Success
  - 1 - No Record Found
  - 2 - Multiple Records Found
  - 3 - Column name conflict with pre-defined tail

  Note: Any errors returned will also appear on the output report.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve the MARGIN,CONTRACT term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1, for Account "ACCT_01" and Product "STANDARD_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59) from the Contract Item Terms table.

```
LOAD_ITEM_TERM.CONTRACTID = "Customer_Pricing_01";
LOAD_ITEM_TERM.REVISION = "1";
LOAD_ITEM_TERM.TERMTYPE = "MARGIN";
LOAD_ITEM_TERM.TERMCATEGORY = "CONTRACT";
LOAD_ITEM_TERM.STARTTIME = "01/01/2008 00:00:00";
LOAD_ITEM_TERM.ACCOUNTID = "ACCT_01";
LOAD_ITEM_TERM.PRODUCTID = "STANDARD_SERVICE";
LOAD_ITEM_TERM.PRODUCTSTART = "01/01/2007 00:00:00";
LOAD_ITEM_TERM.PRODUCTSTOP = "12/31/2010 23:59:59";
ITEM_TERM_DTLS = LOADCONTRACTTERM(LOAD_ITEM_TERM, "ITEM");
```

This function would return the following tail identifiers for the "ITEM_TERM_DTLS" stem identifier:

Tail Identifiers	Value
CONTRACTID	"Customer_Pricing_01"
REVISION	"1"
TERMSTART	"01/01/2006 00:00:00"
TERMSTOP	NULL
TERMTYPE	"MARGIN"
TERMCATEGORY	"CONTRACT"
ACCOUNTID	"ACCT_01"
PRODUCTID	"STANDARD_SERVICE"
PRODUCTSTART	"01/01/2007 00:00:00"
PRODUCTSTOP	"12/31/2010 23:59:59"
STARTTIME	"01/01/2006 00:00:00"
STOPTIME	NULL
VAL	NULL
VALNUM	"5"
VALDATE	NULL

LOADITEMTERMALL Function

Purpose

The LOADITEMTERMALL function loads all contract item terms for a specified contract, contract item, or contract item product from the Oracle Utilities Data Repository. This function can retrieve terms from the Contract Item Term table, the Contract Item Product Term table, or the Contract Item Details table. This function creates one or more stem identifiers containing the retrieved terms. The function returns zero if successful, and returns an integer (1, 2, 3, or 4) if an error occurs.

Format

```
<error_code> = LOADITEMTERMALL(<input_stem>, <table>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the terms to retrieve:
  - CONTRACTID: The contract ID of the contract for the term to be retrieved
  - REVISION: The revision number of the contract for the term to be retrieved
  - ACCOUNTID: The Account ID for the contract item (if applicable)
  - PRODUCTID: The Product ID for the contract item (if applicable)
  - PRODUCTSTART: The Product start time for the contract item (if applicable)
  - PRODUCTSTOP: The Product stop time for the contract item (if applicable)
  - SERVICEPOINT: The Service Point ID for the contract item (if applicable)
  - MARKETID: The Market ID related to the Service Point ID for the contract item (if applicable)
  - SERVICETYPE: The service type related to the Service Point ID for the contract item (if applicable)

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <table> is a string that specifies the table from which the term details are to be retrieved:
  - "ITEM": retrieve the term details from the Contract Item Term table
  - "PRODUCT": retrieve the term details from the Contract Item Product Term table
  - "DETAILS" retrieve the term details from the Contract Item Details table
- <error_code> is the return code of the function call. An error occurs if no term record is found, if multiple terms records are found, if a custom column on the Contract Term table has the same name as one of the pre-defined tail identifiers (see **Term Function Tail Identifiers** on page 7-7), or if the 64-character Rules Language identifier length is exceeded. Return codes are as follows:
  - 0 - Success
  - 1 - No Record Found
  - 2 - Multiple Records Found
  - 3 - Column name conflict with pre-defined tail
  - 4 - Identifier 64-character limit exceeded

  Note: Any errors returned will also appear on the output report.

Stem Identifiers: This function creates one or more stem identifiers that contain the retrieved terms.

- For terms retrieved from the **Contract Item Term** table, stem identifiers are created by concatenating the table name prefix ("ITEM"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  ITEM_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier will be created by concatenating the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  ITEM_MARGIN
  ```

- For terms retrieved from the **Contract Item Product Term** table, stem identifiers are created by concatenating the table name prefix ("IPRD"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  IPRD_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier will be created by concatenating the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  IPRD_MARGIN
  ```

- For terms retrieved from the **Contract Item Details** table, stem identifiers are created by concatenating the table name prefix ("DTLS"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  DTLS_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier will be created by concatenating the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  DTLS_MARGIN
  ```

Stem identifiers are made into array identifiers if there are multiple values of STARTTIME (in the term table) for the same term for the specified contract.

Tail Identifiers: Each stem identifier has the following tail identifiers:

- TERMSTART: The start time of the term
- TERMSTOP: The stop time of the term
- STARTTIME: The start time of the retrieved term
- STOPTIME: The stop time of the retrieved term
- VAL: The text value of the retrieved term
- VALNUM: The numeric value of the retrieved term
- VALDATE: The date value of the retrieved term
- PERIOD: The period for the retrieved term. This tail is only returned when retrieving terms from the Contract Item Details table.
- Any custom columns on the specified table. The tail identifier used will be the same as the custom column name.

See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a value for a column is NULL, the tail identifier will be cleared if it already exists. If the tail identifier that corresponds to a NULL column value does not already exist, it will not be created.

Example

Retrieve all item terms for contact "Customer_Pricing_01", revision 1 for Account "ACCT_01" and Product "STANDARD_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59) from the Contract Item Terms table.

```
LOAD_ALL_ITEM_TERMS.CONTRACTID = "Customer_Pricing_01";
LOAD_ALL_ITEM_TERMS.REVISION = "1";
LOAD_ALL_ITEM_TERMS.ACCOUNTID = "ACCT_01";
LOAD_ALL_ITEM_TERMS.PRODUCTID = "STANDARD_SERVICE";
LOAD_ALL_ITEM_TERMS.PRODUCTSTART = "01/01/2007 00:00:00";
LOAD_ALL_ITEM_TERMS.PRODUCTSTOP = "12/31/2010 23:59:59";
ALL_CONTRACT_ITEM_TERMS = LOADITEMTERMALL(LOAD_ALL_ITEM_TERMS,
"ITEM");
```

This function would return a number of stem identifiers, one for each combination of Term Type and Category. For example:

```
ITEM_MARGIN_CONTRACT
ITEM_DEPOSITAMOUNT_CONTRACT
ITEM_EFFECTIVEPRICESFROM_CONTRACT
ITEM_ONPEAKMARGIN_PRODUCT
ITEM_OFFPEAKMARGIN_PRODUCT
ITEM_CUST_CHARGE_TYPE_PRODUCT
...
```

Each of these stem identifiers would contain the above listed tail identifiers.

If there were multiple Start Time values for the same term, these stem identifiers would be array identifiers, each with an upper bound equal to the number of term records returned.

SAVECONTRACTTERM Function

Purpose

The SAVECONTRACTTERM function saves a single contract term to the Contract Term table in the Oracle Utilities Data Repository. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVECONTRACTTERM(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to save:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
  - TERMSTART: The start time of the term to be saved
  - TERMSTOP: The stop time of the term to be saved
  - TERMTYPE: The term type for the term to be saved
  - TERMCATEGORY: The term category for the term to be saved
  - STARTTIME: The start time of the term to be saved
  - STOPTIME: The stop time of the term to be saved
  - VAL: The text value of the term to be saved
  - VALNUM: The numeric value of the term to be saved
  - VALDATE: The date value of the term to be saved
  - ISSTANDARD: The Is Standard flag of the term to be saved
  - ISREQUIRED: The Is Required flag of the term to be saved
  - ISCALCULATED: The Is Calculated flag of the term to be saved
  - Any custom columns on the Contract Term table. The tail identifier used will be the same as the custom column name.

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save the MARGIN,CONTRACT term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1 with a numeric value of 10.

```
SAVE_TERM.CONTRACTID = "Customer_Pricing_01";
SAVE_TERM.REVISION = "1";
SAVE_TERM.TERMTYPE = "MARGIN";
SAVE_TERM.TERMCATEGORY = "CONTRACT";
SAVE_TERM.STARTTIME = "01/01/2008 00:00:00";
SAVE_TERM.STOPTIME = NULL;
SAVE_TERM.VAL = NULL;
SAVE_TERM.VALNUM = "10";
SAVE_TERM.VALDATE = NULL;
SAVE_TERM.ISSTANDARD = "Yes";
SAVE_TERM.ISREQUIRED = "Yes";
SAVE_TERM.ISCALCULATED = "No";
SAVE_TERM_RETURN = SAVECONTRACTTERM(SAVE_TERM);
```

SAVECONTRACTTERMALL Function

Purpose

The SAVECONTRACTTERMALL function saves all contract terms for a specified contract to the Contract Term table in the Oracle Utilities Data Repository. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVECONTRACTTERMALL(<input_stem>[, <clear_flag>]);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the contract for which terms are to be saved:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
- <clear_flag> is an optional flag that specifies whether or not to clear all stem and tail identifiers associated with the specified contract. A value of "Y" indicates that all stem and tail identifiers be cleared. Any other value indicates that stem and tail identifiers should NOT be cleared.
- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save all terms for contact "Customer_Pricing_01", revision 1 to the Contract Terms table, and clear all associated stem and tail identifiers.

```
SAVE_ALL_TERMS.CONTRACTID = "Customer_Pricing_01";
SAVE_ALL_TERMS.REVISION = "1";
SAVE_ALL_TERMS_RETURN = SAVECONTRACTTERMALL(SAVE_ALL_TERMS, "Y");
```

Notes

This function saves one or more stem identifiers (and their corresponding tail identifiers) that contain previously created or retrieved contract terms (see **LOADCONTRACTTERMALL Function** on page 7-10) based on the contract specified in the function call.

The stem identifiers to be saved are the concatenation of the table name prefix ("CONT"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

```
CONT_MARGIN_CONTRACT
```

If the Category Code is null, then the stem identifier is the concatenation of the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

```
CONT_MARGIN
```

Note: Because the CONTRACTID and REVISION can be different than the contract used to initially create the stem identifiers, not all of the identifiers initially loaded will necessarily be saved.

SAVEGROUPTERM Function

Purpose

The SAVEGROUPTERM function saves a single contract item group term to the Contract Item Group Term table in the Oracle Utilities Data Repository. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVEGROUPTERM(<input_stem>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to save:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
  - TERMSTART: The start time of the term to be saved
  - TERMSTOP: The stop time of the term to be saved
  - TERMTYPE: The term type for the term to be saved
  - TERMCATEGORY: The term category for the term to be saved
  - PRODUCTID: The Product ID for the contract item group
  - PRODUCTSTART: The Product start time for the contract item group
  - PRODUCTSTOP: The Product stop time for the contract item group
  - GROUPID: The Group ID for the contract item group
  - STARTTIME: The start time of the term to be saved
  - STOPTIME: The stop time of the term to be saved
  - VAL: The text value of the term to be saved
  - VALNUM: The numeric value of the term to be saved
  - VALDATE: The date value of the term to be saved
  - Any custom columns on the specified table. The tail identifier used will be the same as the custom column name.

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save the DEPOSITAMOUNT,CONTRACT group term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1 with a numeric value of 100, for group "GROUP_01" and Product "GENERAL_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59) to the Contract Item Group Terms table.

```
SAVE_GROUP_TERM.CONTRACTID = "Customer_Pricing_01";
SAVE_GROUP_TERM.REVISION = "1";
SAVE_GROUP_TERM.TERMTYPE = "DEPOSITAMOUNT";
SAVE_GROUP_TERM.TERMCATEGORY = "CONTRACT";
SAVE_GROUP_TERM.GROUPID = "GROUP_01";
SAVE_GROUP_TERM.PRODUCTID = "GENERAL_SERVICE";
SAVE_GROUP_TERM.PRODUCTSTART = "01/01/2007 00:00:00";
SAVE_GROUP_TERM.PRODUCTSTOP = "12/31/2010 23:59:59";
SAVE_GROUP_TERM.STARTTIME = "01/01/2008 00:00:00";
SAVE_GROUP_TERM.STOPTIME = NULL;
SAVE_GROUP_TERM.VAL = NULL;
SAVE_GROUP_TERM.VALNUM = "100";
SAVE_GROUP_TERM.VALDATE = NULL;
SAVE_GROUP_TERM_RETURN = SAVEGROUPTERM(SAVE_GROUP_TERM);
```

SAVEGROUPTERMALL Function

Purpose

The SAVEGROUPTERMALL function saves all contract group terms for a specified contract to the Contract Item Group Term table in the Oracle Utilities Data Repository. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVEGROUPTERMALL(<input_stem>[, <clear_flag>]);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the contract group for which terms are to be saved:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
  - PRODUCTID: The Product ID for the contract item group
  - PRODUCTSTART: The Product start time for the contract item group
  - PRODUCTSTOP: The Product stop time for the contract item group
  - GROUPID: The Group ID for the contract item group
- <clear_flag> is an optional flag that specifies whether or not to clear all stem and tail identifiers associated with the specified contract group. A value of "Y" indicates that all stem and tail identifiers be cleared. Any other value indicates that stem and tail identifiers should NOT be cleared.
- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save all group terms for contact "Customer_Pricing_01", revision 1, for group "GROUP_01" and Product "GENERAL_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59) to the Contract Item Group Terms table, and clear all associated stem and tail identifiers.

```
SAVE_GROUP_TERMS.CONTRACTID = "Customer_Pricing_01";
SAVE_GROUP_TERMS.REVISION = "1";
SAVE_GROUP_TERMS.PRODUCTID = "GENERAL_SERVICE";
SAVE_GROUP_TERMS.PRODUCTSTART = "01/01/2007 00:00:00";
SAVE_GROUP_TERMS.PRODUCTSTOP = "12/31/2010 23:59:59";
SAVE_GROUP_TERMS.GROUPID = "GROUP_01";
SAVE_GROUP_TERMS_RETURN = SAVECGROUPTERMALL(SAVE_GROUP_TERMS, "Y");
```

Notes

This function saves one or more stem identifiers (and their corresponding tail identifiers) that contain previously created or retrieved group terms (see **LOADGROUPTERMALL Function** on page 7-15) based on the contract group specified in the function call.

The stem identifiers to be saved are the concatenation of the table name prefix ("GRUP"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

```
GRUP_MARGIN_CONTRACT
```

If the Category Code is null, then the stem identifier is the concatenation of the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

```
GRUP_MARGIN
```

> **Note**: Because the CONTRACTID and REVISION can be different than the contract used to initially create the identifiers, not all of the identifiers initially loaded will necessarily be saved.

SAVEITEMTERM Function

Purpose

The SAVEITEMTERM function saves a single contract item term to the Oracle Utilities Data Repository. This function can save terms to the Contract Item Term table, the Contract Item Product Term table, or the Contract Item Details table. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVEITEMTERM(<input_stem>, <table>);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the term to save:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
  - TERMSTART: The start time of the term to be saved
  - TERMSTOP: The stop time of the term to be saved
  - TERMTYPE: The term type for the term to be saved
  - TERMCATEGORY: The term category for the term to be saved
  - ACCOUNTID: The Account ID for the contract item
  - PRODUCTID: The Product ID for the contract item
  - PRODUCTSTART: The Product start time for the contract item
  - PRODUCTSTOP: The Product stop time for the contract item
  - SERVICEPOINT: The Service Point ID for the contract item
  - MARKETID: The Market ID related to the Service Point ID for the contract item
  - SERVICETYPE: The service type related to the Service Point ID for the contract item
  - STARTTIME: The start time of the term to be saved
  - STOPTIME: The stop time of the term to be saved
  - VAL: The text value of the term to be saved
  - VALNUM: The numeric value of the term to be saved
  - VALDATE: The date value of the term to be saved
  - PERIOD: The period for term to be saved. This tail is only returned when saving terms to the Contract Item Details table.
  - Any custom columns on the specified table. The tail identifier used will be the same as the custom column name.

  See **Term Function Tail Identifiers** on page 7-7 for a list of the database table-columns that correspond to these identifiers. If a tail identifier is missing or has previously been cleared (via the CLEAR statement), a NULL value will be used.
- <table> is a string that specifies the table from to the term details are to be saved:
  - "ITEM": save the term to the Contract Item Term table
  - "PRODUCT": save the term to the Contract Item Product Term table
  - "DETAILS": save the term to the Contract Item Details table

- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save the MARGIN,CONTRACT item term with a start date of 01/01/2008 00:00:00 for contact "Customer_Pricing_01", revision 1 with a numeric value of 10, for Account "ACCT_01" and Product "STANDARD_SERVICE" (for 01/01/2007 00:00:00 through 12/31/2010 23:59:59) to the Contract Item Terms table.

```
SAVE_ITEM_TERM.CONTRACTID = "Customer_Pricing_01";
SAVE_ITEM_TERM.REVISION = "1";
SAVE_ITEM_TERM.TERMTYPE = "MARGIN";
SAVE_ITEM_TERM.TERMCATEGORY = "CONTRACT";
SAVE_ITEM_TERM.ACCOUNTID = "ACCT_01";
SAVE_ITEM_TERM.PRODUCTID = "STANDARD_SERVICE";
SAVE_ITEM_TERM.PRODUCTSTART = "01/01/2007 00:00:00";
SAVE_ITEM_TERM.PRODUCTSTOP = "12/31/2010 23:59:59";
SAVE_ITEM_TERM.STARTTIME = "01/01/2008 00:00:00";
SAVE_ITEM_TERM.STOPTIME = NULL;
SAVE_ITEM_TERM.VAL = NULL;
SAVE_ITEM_TERM.VALNUM = "10";
SAVE_ITEM_TERM.VALDATE = NULL;
SAVE_ITEM_TERM_RETURN = SAVEITEMTERM(SAVE_ITEM_TERM,"ITEM");
```

SAVEITEMTERMALL Function

Purpose

The SAVEITEMTERMALL function saves all contract item terms for a specified contract to the Oracle Utilities Data Repository. This function can save terms to the Contract Item Term table, the Contract Item Product Term table, or the Contract Item Details table. The function returns zero (0) if successful, and 1 if an error occurs.

Format

```
<return_code> = SAVEITEMTERMALL(<input_stem>, <table>[,
<clear_flag>]);
```

Where

- <input_stem> is a stem identifier with following tail identifiers that specify the contract and contract item for which terms are to be saved:
  - CONTRACTID: The contract ID of the contract for the term to be saved
  - REVISION: The revision number of the contract for the term to be saved
  - ACCOUNTID: The Account ID for the contract item
  - PRODUCTID: The Product ID for the contract item
  - PRODUCTSTART: The Product start time for the contract item
  - PRODUCTSTOP: The Product stop time for the contract item
  - SERVICEPOINT: The Service Point ID for the contract item
  - MARKETID: The Market ID related to the Service Point ID for the contract item
  - SERVICETYPE: The service type related to the Service Point ID for the contract item
- <table> is a string that specifies the table to which the term details are to be saved:
  - "ITEM": save the term to the Contract Item Term table
  - "PRODUCT": save the term to the Contract Item Product Term table
  - "DETAILS": save the term to the Contract Item Details table
- <clear_flag> is an optional flag that specifies whether or not to clear all stem and tail identifiers associated with the specified contract. A value of "Y" indicates that all stem and tail identifiers be cleared. Any other value indicates that stem and tail identifiers should NOT be cleared.
- <return_code> is the return code of the function call. An error occurs if the term record cannot be saved. Return codes are as follows:
  - 0 - Success
  - 1 - Record could not be saved

  Note: Any errors returned will also appear on the output report.

Example

Save all terms for contact "Customer_Pricing_01", revision 1 to the Contract Terms table, and clear all associated stem and tail identifiers.

```
SAVE_ALL_TERMS.CONTRACTID = "Customer_Pricing_01";
SAVE_ALL_TERMS.REVISION = "1";
SAVE_ALL_TERMS_RETURN = SAVECONTRACTTERMALL(SAVE_ALL_TERMS, "ITEM",
"Y");
```

Notes

This function saves one or more stem identifiers (and their corresponding tail identifiers) that contain previously created or retrieved contract item terms (see **LOADITEMTERMALL Function** on page 7-20) based on the contract and contract item specified in the function call.

> **Note**: Because the CONTRACTID and REVISION can be different than the contract used to initially create the identifiers, not all of the identifiers initially loaded will necessarily be saved.

- For terms to be saved to the **Contract Item Term** table, stem identifiers are the concatenation of the table name prefix ("ITEM"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  ITEM_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier is the concatenation of the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  ITEM_MARGIN
  ```

- For terms to be saved to the **Contract Item Product Term** table, stem identifiers are the concatenation of the table name prefix ("IPRD"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  IPRD_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier is the concatenation of the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  IPRD_MARGIN
  ```

- For terms to be saved to the **Contract Item Details** table, stem identifiers are the concatenation of the table name prefix ("DTLS"), an underscore, the Term Type Code, an underscore, and the Category Code. For example, the stem identifier for contract terms of type "MARGIN" of category "CONTRACT" would be as follows:

  ```
  DTLS_MARGIN_CONTRACT
  ```

  If the Category Code is null, then the stem identifier is the concatenation of the table name prefix, an underscore, and the Term Type Code. For example, the stem identifier for contract terms of type "MARGIN" with a Null category would be as follows:

  ```
  DTLS_MARGIN
  ```

Customizing Oracle Utilities Quotations Management Calculations and Contracts

You can also customize the **Contract Calculations** report (opened via the **View Calculations** action on the **Actions** drop-down list on the Contract screen), and the **Contract** itself (opened via the **View Contract** action on the **Actions** drop-down list on the Contract screen).

Both of these functions use a report template (supplied with your installation of Oracle Utilities Quotations Management) that should be customized to suit your business needs. These report templates are located in the **C:\LODESTAR\BIP\Reports\Shared\CE** directory on the web server where you installed the web-enabled Oracle Utilities Quotations Management application.

Oracle Utilities Quotations Management Report Templates

The two report templates used by Oracle Utilities Quotations Management are outlined below.

> **Note**: The sample templates provided by Oracle Utilities are place-holders, and contain no business logic or formatting of any kind, and must be customized before using the Oracle Utilities Quotations Management application.

Contract Calculations Report

Report Template: ViewCalculations

Purpose: Used to report contract calculations, based on user-defined criteria

Contract

Report Template: ViewContract

Purpose: Used to create the contract document, which can be printed or exported to be sent to the appropriate counter parties.

Customizing the Contracts Screen

You can also customize the appearance and behavior of the Contracts screen, including:

- **Customizing Contract Tabs**
- **Customizing the Actions Menu**
- **Creating Custom Approval Processes**

Before you make any custom changes to the Contracts screen, review the **Customization Ground Rules** on page 10-3 in the *Oracle Utilities Energy Information Platform Configuration Guide*.

Customizing Contract Tabs

Customizing the tabs on the Contracts screen allows you to change the order in which the tabs appear, or to add custom tabs.

To customize the tabs on the Contracts screen, use the following steps:

1. Create a custom folder (such as "zcustom") in the **C:\LODESTAR\Web\classic** folder. This is the folder where all your custom files should be located.
2. Create a copy of the "ContractTabs.xml" file in the custom folder. This file defines your customization. See **"ContractTabs.XML" File Structure** on page 7-35 for details about creating this file. This file is located in the C:\LODESTAR\Web\classic\cm folder on the web server.
3. Edit the copy of the "ContractTabs.xml" file as appropriate for your customizations.
4. Create a text file called "dictionary.dic" in the custom folder. This file defines the labels for the custom menu items and other custom content. See **"Dictionary.dic" File Structure** on page 7-38 for details about creating this file.

Editing the ContractTabs.XML File

The "ContractTabs.xml" file defines the tabs that appear on the Contracts screen.

"ContractTabs.XML" File Structure

The basic structure of the "ContractTabs.xml" file is as follows:

```
<tabs>
  <item name="[TAB_NAME]" caption="[ORDER]" action="[ACTION]"
secure="[SECURITY_NODE]" include="[STATUS_LIST]"
includeext="[EXT_STATUS_LIST]" includecat="[CATEGORY_LIST]"
exclude="[STATUS_LIST]" excludeext="[EXT_STATUS_LIST]"
excludecat="[CATEGORY_LIST]"/>
</tabs>
```

where:

tabs: is the root element of the "ContractTabs.xml" file, and can contain one or more items.

item: is an element that defines an individual menu item. Item elements have the following attributes:

- **name**: is the name of an individual tab (BASICS, ITEMS, TERMS, etc.). Item names **cannot** have spaces in them.
- **caption**: A number that designates the order in which the tabs appear on the screen, from left to right. The left-most tab is numbered "1."
- **action**: is the path and file name of the ASP file for the tab, in the following format:

  ```
  ../<path>/<filename>
  ```

  where:

    - <path> is the path (relative to the C:\LODESTAR folder) to the file to be opened by the menu item.
    - <filename> is the name of the file to be opened by the menu item.
- **include**: a space-separated list of Contract Status Codes. The tab is enabled if the current Contract Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: include=" INPROCESS INAPPROVAL ")
- **includeext**: a space-separated list of External Contract Status Codes. The tab is enabled if the current Contract External Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: includeext=" INPROCESS ")
- **includecat**: a space-separated list of Contract Category Codes. The tab is enabled if the current Contract category code matches one of the codes specified. Note: This list must begin and end with a space (ex: includecat=" CONTRACT ")
- **exclude**: a space-separated list of Contract Status Codes. The tab is disabled if the current Contract Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: exclude=" INPROCESS INAPPROVAL ")
- **excludeext**: a space-separated list of External Contract Status Codes. The tab is disabled if the current Contract External Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: excludeext=" INPROCESS ")
- **excludecat**: a space-separated list of Contract Category Codes. The tab is disabled if the current Contract category code matches one of the codes specified. Note: This list must begin and end with a space (ex: excludecat=" CONTRACT ")

Default ContractTabs.XML

The "default" version of the "ContractTabs.xml" file (located in the C:\LODESTAR\Web\classic\cm folder on the web server) is as follows:

```
<tabs>
  <item name="BASICS" caption="1" action="../cm/ContractBasics.asp" secure="//
CONTRACTTABS/BASICS"/>
  <item name="ITEMS" caption="2" action="../cm/ContractItems.asp" secure="//
CONTRACTTABS/ITEMS"/>
  <item name="TERMS" caption="3" action="../cm/ContractTerms.asp" secure="//
CONTRACTTABS/TERMS"/>
  <item name="DOCUMENTS" caption="4" action="../cm/ContractDocuments.asp"
secure="//CONTRACTTABS/DOCUMENTS"/>
  <item name="APPROVALS" caption="5" action="../cm/ContractApprovals.asp"
secure="//CONTRACTTABS/APPROVALS"/>
  <item name="HISTORY" caption="6" action="../cm/ContractHistory.asp" secure="/
/CONTRACTTABS/HISTORY"/>
  <item name="RELATED" caption="7" action="../cm/ContractRelated.asp" secure="/
/CONTRACTTABS/RELATED"/>
  <item name="WQDETAIL" caption="8" action="../cm/ContractWQ.asp" secure="//
CONTRACTTABS/WQDETAIL"/>
</tabs>
```

Changing the order of the tabs

To customize the order in which the tabs appear on the Contracts screen, change the values of the "caption" attributes accordingly in the copy of the "ContractTabs.XML" file. For example, the sample file below would changes the order of the tabs to be as follows:

Basics, Terms, Items, Documents, Approvals, Related Contracts, Work Queue Details, and History

```
<tabs>
  <item name="BASICS" caption="1" action="../cm/ContractBasics.asp" secure="//
CONTRACTTABS/BASICS"/>
  <item name="ITEMS" caption="3" action="../cm/ContractItems.asp" secure="//
CONTRACTTABS/ITEMS"/>
  <item name="TERMS" caption="2" action="../cm/ContractTerms.asp" secure="//
CONTRACTTABS/TERMS"/>
```

```
  <item name="DOCUMENTS" caption="4" action="../cm/ContractDocuments.asp"
secure="//CONTRACTTABS/DOCUMENTS"/>
  <item name="APPROVALS" caption="5" action="../cm/ContractApprovals.asp"
secure="//CONTRACTTABS/APPROVALS"/>
  <item name="HISTORY" caption="8" action="../cm/ContractHistory.asp" secure="/
/CONTRACTTABS/HISTORY"/>
  <item name="RELATED" caption="6" action="../cm/ContractRelated.asp" secure="/
/CONTRACTTABS/RELATED"/>
  <item name="WQDETAIL" caption="7" action="../cm/ContractWQ.asp" secure="//
CONTRACTTABS/WQDETAIL"/>
</tabs>
```

Adding Custom Tabs

To add custom tabs to the Contracts screen, add an "item" element to the copy of the "ContractTabs.xml" file. For example, the sample file below would add a tab called "CUSTOM" that opens the "example1.asp" file in the "zContracts" folder.

```
<tabs>
  <item name="CUSTOM" caption="9" action="../zContracts/example1.asp"/>
</tabs>
```

To add a custom tab and change the order in which the tabs appear, you could combine the two previous sample files as follows:

```
<tabs>
  <item name="BASICS" caption="1" action="../cm/ContractBasics.asp" secure="//
CONTRACTTABS/BASICS"/>
  <item name="ITEMS" caption="3" action="../cm/ContractItems.asp" secure="//
CONTRACTTABS/ITEMS"/>
  <item name="TERMS" caption="2" action="../cm/ContractTerms.asp" secure="//
CONTRACTTABS/TERMS"/>
  <item name="DOCUMENTS" caption="4" action="../cm/ContractDocuments.asp"
secure="//CONTRACTTABS/DOCUMENTS"/>
  <item name="APPROVALS" caption="5" action="../cm/ContractApprovals.asp"
secure="//CONTRACTTABS/APPROVALS"/>
  <item name="HISTORY" caption="8" action="../cm/ContractHistory.asp" secure="/
/CONTRACTTABS/HISTORY"/>
  <item name="RELATED" caption="6" action="../cm/ContractRelated.asp" secure="/
/CONTRACTTABS/RELATED"/>
  <item name="WQDETAIL" caption="7" action="../cm/ContractWQ.asp" secure="//
CONTRACTTABS/WQDETAIL"/>
  <item name="CUSTOM" caption="9" action="../zContracts/example1.asp"/>
</tabs>
```

Changing Availability of Tabs

To change which tabs are accessible based on a specific Status code, External Status code, or Category code, add/edit the "include", "includeext", "includecat", "exclude", "excludeext", or "excludecat" attribute in the copy of the "ContractTabs.xml" file. These attributes specify which Status Codes, External Status Codes, or Contract Category Codes that enable or disable each tab.

For example, the sample file below would enable the "CUSTOM" tab when the contract is in one of the following statuses: In Process, In Approval, or Submitted. The "CUSTOM" tab would be disabled when in any other status.

```
<tabs>
  <item name="CUSTOM" caption="9" action="../zContracts/example1.asp" include="
INPROCESS INAPPROVAL SUBMITTED"/>
</tabs>
```

Creating the Dictionary File

Dictionary files define the labels that will be visible on the user interface for the custom tabs defined in the "ContractTabs.xml" file. Dictionary files also define labels from other files, including custom ASP or HTML pages you might develop.

Dictionary files must be located in the same folder as the "ContractTabs.xml" file (or other custom files).

Without proper dictionary files in place, all item names will be appear in brackets ({ }).

Note: When you create or update a "dictionary.dic" file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

"Dictionary.dic" File Structure

The basic structure of the "dictionary.dic" file, when used with contract tabs, is as follows:

```
<dictionary>
  <CM>
    <CUSTOM ENU="Custom Tab"/>
  </CM>
</dictionary>
```

where:

dictionary: is the root element of the "dictionary.dic" file.

CM: is an element which contains labels for the Contract screens.

<NAME>: is an element that corresponds to the "name" attribute of a menu section, division, or item. In the above example, "CUSTOM" is the name of the custom tab. <NAME> elements have the following attributes:

- **<LANGUAGE>**: a three letter ISO code that designates the language of the element. ENU is English, and is the default. When creating dictionary files or entries for other languages, the appropriate ISO code is used. For example, for French dictionary files, you would use "FRA". For Italian dictionary files you would use "ITA".

Example

To create a label for the custom tab in the above example, you would create the following in the "dictionary.dic" file.

```
<dictionary>
  <CM>
    <CUSTOM ENU="Custom Tab"/>
  </CM>
</dictionary>
```

In this example, the custom tab would be labeled "Custom Tab."

Securing Tabs

Securing tabs allows you to restrict which users (or groups of users) can access your custom tabs.

To secure custom tabs, use the following steps:

1. Add security nodes to the "ContractTabs.xml" file. These nodes define the security privileges associated with your custom tabs. See **"ContractTabs.XML" File Structure - Security Nodes** on page 7-39 for details about editing this file to add security nodes.
2. Create a permissions schema file in the **C:\LODESTAR\CFG** folder. This file defines the security nodes used to restrict access to your custom tabs. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface. See **"Permission.*.XSD" File Structure** on page 7-40 for details about creating this file.

3. Create a permissions dictionary file in the **C:\LODESTAR\CFG** folder. This file defines the labels for the security nodes that will appear on the Security Administration user interface. See **Creating Permissions Dictionary Files** on page 10-11 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about creating Permissions Dictionary files.

Adding Security Nodes to the "ContractTabs.XML" File

Adding security nodes to he "ContractTabs.xml" file defines the security privileges associated with your custom tabs and enables securing those custom tabs on the Energy Information Platform user interface.

"ContractTabs.XML" File Structure - Security Nodes

The structure of the "ContractTabs.xml" file changes slightly when securing menu items, as follows (changes noted in bold):

```
<tabs>
  <item name="CUSTOM" caption="9" action="../zContracts/example1.asp" secure="/
/CONTRACTTABS/CUSTOM"/>
</tabs>
```

where:

secure: is an attribute on an item that specifies the security nodes associated with the tab. The value specified for this attribute represents the xml structure in the Permissions schema (*.XSD) file used to define the security nodes associated with the tabs (see **Creating Permissions Schema Files** on page 7-40 for more information about permission schema files). In the above example, CUSTOM is a node of CONTRACTTABS (which is a node of PERMISSIONS). The format for this attribute is as follows:

```
//<section_node>[/<item_node>]
```

where:

- <section_node> is the name of the security node associated with the tabs on the Contracts screen.

  Example: **`//CONTRACTTABS`**

- <item_node> is the name of the security node associated with the tab.

  Example: **`//CONTRACTTABS/CUSTOM`**

Note: Nodes must be in UPPER case.

Example

To add security nodes to the custom tab described, you would edit the "ContractTabs.xml" file as follows (changed noted in **bold**).

```
<tabs>
  <item name="CUSTOM" caption="9" action="../zContracts/example1.asp" secure="/
/CONTRACTTABS/CUSTOM"/>
</tabs>
```

In this example, the top level node is called "CONTRACTTABS." Beneath that would be a node called "CUSTOM".

Creating Permissions Schema Files

Permissions schemas are hierarchical XML structures that define the security nodes used to restrict access to your custom menu items. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface.

Permissions schema files use the following naming convention:

permissions.<CUSTOM_NAME>.xsd

where:

- <CUSTOM_NAME> is a unique name that identifies the schema file.

Note: When you create or update a "permissions.*.xsd" file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

"Permission.*.XSD" File Structure

The structure of the permission schema file is as follows:

```
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="CONTRACTTABS" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="[TAB_NODE]" minOccurs="0"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

where:

PERMISSIONS: is the parent node. All security nodes are children of the PERMISSIONS node.

CONTRACT: is the node that corresponds to the Contract screen.

CONTRACTTABS: is the node that corresponds to the Contract tabs section in the "ContractTabs.xml" file.

TAB_NODE: is a node that corresponds to a tab in the "ContractTabs.xml" file.

How to create a permissions schema file:

To create a permissions schema file, use the following steps.

1. Create a text file in the C:\LODESTAR\CFG folder using the following naming scheme:

   permissions.<CUSTOM_NAME>.xsd

   where:

   - **<CUSTOM_NAME>** is a unique name

2. Copy the contents of the **permissions.500_datasources.xsd** file from the C:\LODESTAR\Bin folder into this new file. The contents of this file are as follows:

```
<xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="DATASOURCE" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3. Change the DATASOURCE node to "CONTRACT." This is the parent node of the CONTRACTTABS node.

```
<xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. Add a CONTRACTTABS node (or nodes) under the CONTRACT node element. This element should use the following syntax:

```
...
    <xs:element name="CONTRACT" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="CONTRACTTABS" minOccurs="0">
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
...
```

5. Add a node (or nodes) for each custom tab under the CONTRACTTABS node element. This must be the exact division node as specified in the "ContractTabs.xml" file ("CUSTOM"). This element should use the following syntax:

```
...
    <xs:element name="CONTRACT" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="CONTRACTTABS" minOccurs="0">
            <xs:complexType>
              <xs:all>
                <xs:element name="CUSTOM" minOccurs="0"/>
              </xs:all>
            </xs:complexType>
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
...
```

Example

The permission schema file (permissions.custom.xsd) for the custom tab described above would be as follows (nodes denoted in **bold**):

```
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="CONTRACTTABS" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="CUSTOM" minOccurs="0"/>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Creating Permissions Dictionary Files

Just as menu items used dictionary files to define the labels that will be visible on the user interface, security nodes defined in permission schema files need dictionary files to define the labels for the security nodes that will appear on the Security Administration user interface.

Permission dictionary files must be located in the C:\LODESTAR\CFG folder.

Without proper dictionary files in place, all security nodes, including nodes for sections, divisions, and items will be appear in brackets ({ }). For instance, if you didn't create a permissions dictionary.dic file for the permissions schema described above, the privileges would look like the following in the Feature Privileges tree on the Security Administration the user interface:

Example Permissions Dictionary File - Custom Tab

To create labels for the security nodes in the above custom tab example, you would create the following file ("permissions.custom.dic"):

```
<dictionary>
  <security>
    <PERMISSIONS ENU="Features">
      <CONTRACT ENU="Contract Management">
        <CONTRACTTABS ENU="Contract Tab Set">
          <CUSTOM ENU="Custom Tab"/>
        </CONTRACTTABS>
      </CONTRACT>
    </PERMISSIONS>
  </security>
</dictionary>
```

See **Creating Permissions Dictionary Files** on page 10-11 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about creating Permissions Dictionary files.

Customizing the Actions Menu

Customizing the Actions menu on the Contracts screen allows you to change the actions available on the Actions drop-down menu. Actions on this menu can invoke Oracle Utilities Rules Language processing to perform custom business logic.

To customize the Actions menu on the Contracts screen, use the following steps:

1. Create a custom folder (such as "zcustom") in the **C:\LODESTAR\Web\classic** folder. This is the folder where all your custom files should be located.
2. Create a copy of the "ContractMenu.xml" file in the custom folder. This file defines your custom changes to the Actions menu. See **"ContractMenu.XML" File Structure** on page 7-43 for details about creating these files. This file is located in the C:\LODESTAR\Web\classic\cm folder on the web server.
3. Create a copy of the "ContractRates.xml" file in the custom folder. Thus file defines the rate schedule executed by your custom actions on the Actions menu. See **"ContractRates.XML" File Structure** on page 7-47 for details about creating this file. This file is located in the C:\LODESTAR\Web\classic\cm folder on the web server.
4. Edit the copies of the "ContractMenu.xml" and "ContractRates.xml" files as appropriate for your customizations.
5. Create a text file called "dictionary.dic" in the custom folder. This file defines the labels for the custom actions. See **"Dictionary.dic" File Structure** on page 7-38 for details about creating this file.

Editing the ContractMenu.XML File

The "ContractMenu.xml" file defines the actions that appear on the Actions menu on the Contracts screen.

"ContractMenu.XML" File Structure

The basic structure of the "ContractMenu.xml" file is as follows:

```
<menubar>
  <menu name="Actions">
    <item name="[ACTION]"  param="[PARAM]" action="../cm/DropDownActions.asp"
icon="[ICON]" secure="[SECURITY_NODE]"  include="[STATUS_LIST]"
includeext="[EXT_STATUS_LIST]" includecat="[CATEGORY_LIST]"
exclude="[STATUS_LIST]" excludeext="[EXT_STATUS_LIST]"
excludecat="[CATEGORY_LIST]" />
  </menu>
</menubar>
```

where:

menubar: is the root element of the "ContractMenu.xml" file, that contain one or more menus.

menu: defines a menu. Can contain one or more item elements.

item: is an element that defines an individual menu item on the Actions menu. An empty <item> element (<item/>) inserts a line in the Actions menu. This can be used to delineate specific sections of the menu. Item elements have the following attributes:

- **name**: is the name of an individual action (Save, SaveComments, CalculateTerms, etc.). Item names **cannot** have spaces in them.
- **param**: name used
- **action**: is the path and file name of the ASP file for the tab, in the following format:
- **icon**: name of the icon to display on the Actions menu. Valid values include: "save", "csv", "approval", "reject", "submit2", "Exec", "stop", and "restore".

- **include**: a space-separated list of Contract Status Codes. The action is enabled if the current Contract Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: include=" INPROCESS INAPPROVAL ")
- **includeext**: a space-separated list of External Contract Status Codes. The action is enabled if the current Contract External Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: includeext=" INPROCESS ")
- **includecat**: a space-separated list of Contract Category Codes. The action is enabled if the current Contract category code matches one of the codes specified. Note: This list must begin and end with a space (ex: includecat=" CONTRACT ")
- **exclude**: a space-separated list of Contract Status Codes. The action is disabled if the current Contract Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: exclude=" INPROCESS INAPPROVAL ")
- **excludeext**: a space-separated list of External Contract Status Codes. The action is disabled if the current Contract External Status code matches one of the codes specified. Note: This list must begin and end with a space (ex: excludeext=" INPROCESS ")
- **excludecat**: a space-separated list of Contract Category Codes. The action is disabled if the current Contract category code matches one of the codes specified. Note: This list must begin and end with a space (ex: excludecat=" CONTRACT ")

Default ContractMenu.XML

The "default" version of the "ContractMenu.xml" file (located in the C:\LODESTAR\Web\classic\cm folder on the web server) is as follows:

```
<menubar>
  <menu name="Actions">
    <item name="Save"  param="Save" action="../cm/DropDownActions.asp"
icon="save" secure="//CONTRACTTABS/ACTIONS/@UPDATE"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED " />
    <item name="SaveComments" param="SaveAs" prompt="Comments" action="../cm/
DropDownActions.asp" icon="save" secure="//CONTRACTTABS/ACTIONS/@UPDATE"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED " />
    <item/>
    <item name="CalculateTerms" param="CalculateTerms" action="../cm/
DropDownActions.asp" icon="csv" secure="//CONTRACTTABS/ACTIONS/
@CALCULATETERMS"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REVISED "/>
    <item name="CalculatePricing" param="CalculatePricing" action="../cm/
DropDownActions.asp" icon="csv" target="_blank" secure="//CONTRACTTABS/ACTIONS/
@VIEWCALCULATIONS"/>
    <item name="ViewContract" param="ViewContract" action="../cm/
DropDownActions.asp" icon="csv" target="_blank" secure="//CONTRACTTABS/ACTIONS/
@VIEWCONTRACT"/>
    <item/>
    <item name="ReadyForApproval" param="ContractApprovals" action="../cm/
DropDownActions.asp" icon="approval" secure="//CONTRACTTABS/ACTIONS/
@READYFORAPPROVAL"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED "
      excludeext=" AWAITINGRISK READYTOSUBMIT SUBMITTED AWAITINGCREDIT
READYTOEXECUTE EXECUTED REJECTED REVISED TERMINATED " />
    <item name="Approve"  param="Approve" icon="approval" action="../cm/
DropDownActions.asp" secure="//CONTRACTTABS/APPROVALS/APPROVALEX"
      isapproval="true"
      include="INAPPROVAL"
      excludeext=" INPROCESS READYTOSUBMIT SUBMITTED READYTOEXECUTE EXECUTED
REJECTED REVISED TERMINATED " />
    <item name="Reject" param="Reject" icon="reject" action="../cm/
DropDownActions.asp" secure="//CONTRACTTABS/APPROVALS/APPROVALEX"
```

```
      isapproval="true"
      include="INAPPROVAL"
      excludeext=" INPROCESS READYTOSUBMIT SUBMITTED READYTOEXECUTE EXECUTED
REJECTED REVISED TERMINATED "/>
    <item name="SubmitContract" param="SubmitContract" action="../cm/
DropDownActions.asp" icon="submit2" secure="//CONTRACTTABS/ACTIONS/
@SUBMITCONTRACT"
      exclude=" INPROCESS SUBMITTED EXECUTED TERMINATED ARCHIVED REJECTED
REVISED "
      excludeext=" INPROCESS AWAITINGRISK SUBMITTED AWAITINGCREDIT
READYTOEXECUTE EXECUTED REJECTED REVISED TERMINATED "/>
    <item name="ExecuteContract" param="ExecuteContract" action="../cm/
DropDownActions.asp" icon="Exec" secure="//CONTRACTTABS/ACTIONS/
@EXECUTECONTRACT"
      exclude=" INPROCESS INAPPROVAL EXECUTED TERMINATED ARCHIVED REJECTED
REVISED "
      excludeext=" INPROCESS AWAITINGRISK READYTOSUBMIT SUBMITTED
AWAITINGCREDIT EXECUTED REJECTED REVISED TERMINATED "/>
    <item name="TerminateContract" param="TerminateContract" action="../cm/
DropDownActions.asp" icon="stop" secure="//CONTRACTTABS/ACTIONS/
@TERMINATECONTRACT"
      exclude=" TERMINATED ARCHIVED REJECTED REVISED "
      excludeext=" AWAITINGRISK AWAITINGCREDIT REJECTED REVISED TERMINATED "/>
    <item name="ReviseContract" param="ReviseContract" action="../cm/
DropDownActions.asp" icon="restore" secure="//CONTRACTTABS/ACTIONS/
@REVISECONTRACT"
      exclude=" REVISED "
      excludeext=" TERMINATED ARCHIVED REJECTED REVISED "/>
  </menu>
</menubar>
```

Changing the order of actions

To customize the order in which the actions appear on the Actions menu, change the order of the <item> elements accordingly in the copy of the "ContractMeanu.XML" file. For example, the sample file below would changes the order of the actions to be as follows:

Save w/Comments, Save, Calculate Pricing, View Contract, Calculate Terms, Ready for Approval, Submit Contract, Execute Contract, Terminate Contract, and Revise Contract.

```
<menubar>
  <menu name="Actions">
    <item name="SaveComments" param="SaveAs" prompt="Comments" action="../cm/
DropDownActions.asp" icon="save" secure="//CONTRACTTABS/ACTIONS/@UPDATE"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED " />
    <item name="Save"  param="Save" action="../cm/DropDownActions.asp"
icon="save" secure="//CONTRACTTABS/ACTIONS/@UPDATE"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED " />
    <item/>
    <item name="CalculatePricing" param="CalculatePricing" action="../cm/
DropDownActions.asp" icon="csv" target="_blank" secure="//CONTRACTTABS/ACTIONS/
@VIEWCALCULATIONS"/>
    <item name="ViewContract" param="ViewContract" action="../cm/
DropDownActions.asp" icon="csv" target="_blank" secure="//CONTRACTTABS/ACTIONS/
@VIEWCONTRACT"/>
    <item name="CalculateTerms" param="CalculateTerms" action="../cm/
DropDownActions.asp" icon="csv" secure="//CONTRACTTABS/ACTIONS/
@CALCULATETERMS"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REVISED "/>
    <item/>
    <item name="ReadyForApproval" param="ContractApprovals" action="../cm/
DropDownActions.asp" icon="approval" secure="//CONTRACTTABS/ACTIONS/
@READYFORAPPROVAL"
      exclude=" INAPPROVAL APPROVED SUBMITTED EXECUTED TERMINATED ARCHIVED
REJECTED REVISED "
```

```
      excludeext=" AWAITINGRISK READYTOSUBMIT SUBMITTED AWAITINGCREDIT
READYTOEXECUTE EXECUTED REJECTED REVISED TERMINATED " />
    <item name="SubmitContract" param="SubmitContract" action="../cm/
DropDownActions.asp" icon="submit2" secure="//CONTRACTTABS/ACTIONS/
@SUBMITCONTRACT"
      exclude=" INPROCESS SUBMITTED EXECUTED TERMINATED ARCHIVED REJECTED
REVISED "
      excludeext=" INPROCESS AWAITINGRISK SUBMITTED AWAITINGCREDIT
READYTOEXECUTE EXECUTED REJECTED REVISED TERMINATED "/>
    <item name="ExecuteContract" param="ExecuteContract" action="../cm/
DropDownActions.asp" icon="Exec" secure="//CONTRACTTABS/ACTIONS/
@EXECUTECONTRACT"
      exclude=" INPROCESS INAPPROVAL EXECUTED TERMINATED ARCHIVED REJECTED
REVISED "
      excludeext=" INPROCESS AWAITINGRISK READYTOSUBMIT SUBMITTED
AWAITINGCREDIT EXECUTED REJECTED REVISED TERMINATED "/>
    <item name="TerminateContract" param="TerminateContract" action="../cm/
DropDownActions.asp" icon="stop" secure="//CONTRACTTABS/ACTIONS/
@TERMINATECONTRACT"
      exclude=" TERMINATED ARCHIVED REJECTED REVISED "
      excludeext=" AWAITINGRISK AWAITINGCREDIT REJECTED REVISED TERMINATED "/>
    <item name="ReviseContract" param="ReviseContract" action="../cm/
DropDownActions.asp" icon="restore" secure="//CONTRACTTABS/ACTIONS/
@REVISECONTRACT"
      exclude=" REVISED "
      excludeext=" TERMINATED ARCHIVED REJECTED REVISED "/>
  </menu>
</menubar>
```

Adding Actions

To add custom actions to the Contracts screen, define an "item" element to the copy of the "ContractMenu.xml" file. For example, the sample file below would add an action called "CUSTOM" to the Actions menu that would be available for all Status Codes, External Status Codes, and Contract Categories.

```
<menubar>
  <menu name="Actions">
    <item name="CUSTOM" param="CUSTOM" prompt="Custom" action="../cm/
DropDownActions.asp" icon="save" />
  </menu>
</menubar>
```

Changing Availability of Actions

To change which Actions are accessible based on a specific Status code, External Status code, or Category code, add/edit the "include", "includeext", "includecat", "exclude", "excludeext", or "excludecat" attribute in the copy of the "ContractMenu.xml" file. These attributes specify which Status Codes, External Status Codes, or Contract Category Codes that enable or disable each action.

For example, the sample file below would enable the "CUSTOM" action when the contract is in one of the following statuses: In Process, In Approval, or Submitted. The "CUSTOM" action would be disabled when in any other status.

```
<menubar>
  <menu name="Actions">
    <item name="CUSTOM" param="CUSTOM" prompt="Custom" action="../cm/
DropDownActions.asp" icon="save"  include=" INPROCESS INAPPROVAL SUBMITTED"/>
  </menu>
</menubar>
```

Editing the ContractRates.XML File

The "ContractRates.xml" file specifies the Oracle Utilities Rules Language rate schedules executed by the actions that appear on the Actions menu on the Contracts screen.

"ContractRates.XML" File Structure

The basic structure of the "ContractRates.xml" file is as follows:

```
<RATE_SCHEDULES>
  <RATE NAME="[PARAM]" RATE_CODE="[RATE_CODE]" OPERATING_COMPANY="[OPCO]"
JURISDICTION="[JURIS]">
    <RATE_INPUT NAME="[RATE_IDENTIFIER]" TYPE="[ID_TYPE]"/>
  </RATE>
</RATE_SCHEDULES>
```

where:

RATE_SCHEDULES: is the root element of the "ContractRates.xml" file, that contain one or more <RATE> elements.

RATE: is an element that defines the rate schedule to be executed by the action on the Actions menu. RATE elements have the following attributes:

- **NAME**: is the value of the "param" attribute of the action, as defined in the "ContractMenu.xml" file.
- **RATE_CODE**: is the Rates Schedule Code of the rate schedule to be executed by the action.
- **OPERATING_COMPANY**: is the Operating Company Code of the rate schedule to be executed by the action.
- **JURISDICTION**: is the Jurisdiction Code of the rate schedule to be executed by the action.

RATE_INPUT: is an element that defines an individual input identifier to be passed to the rate schedule. RATE_INPUT elements have the following attributes:

- **NAME**: is the name of the Rules Language identifier passed. Names **cannot** have spaces in them. Since more actions will be based on the Contract ID and Revision of the current contract, most RATE elements have RATE_INPUT elements for both of these.
- **TYPE**: The data type of the Rules Language identifier passed. Valid values include "S" (string), "I" (integer), or "D" (date).

Default ContractRates.XML

The "default" version of the "ContractRates.xml" file (located in the C:\LODESTAR\Web\classic\cm folder on the web server) is as follows:

```
<RATE_SCHEDULES>
  <RATE NAME="CreateContract" RATE_CODE="LSCM_CREATE_CONTRACT"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CUSTOMER_ACCOUNT_FLG" TYPE="S"/>
    <RATE_INPUT NAME="INPUT_LIST" TYPE="S"/>
    <RATE_INPUT NAME="INPUT_ID" TYPE="S"/>
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="CONTRACT_DESCRIPTION" TYPE="S"/>
    <RATE_INPUT NAME="CONTRACT_TYPECODE" TYPE="S"/>
    <RATE_INPUT NAME="CONTRACT_CATEGORYCODE" TYPE="S"/>
    <RATE_INPUT NAME="PARENT_CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="CONTRACT_STARTTIME" TYPE="D"/>
    <RATE_INPUT NAME="CONTRACT_STOPTIME" TYPE="D"/>
    <RATE_INPUT NAME="CONTRACTEE_ID" TYPE="S"/>
  </RATE>
  <RATE NAME="CalculatePricing" RATE_CODE="LSCM_CALCULATE_PRICE"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
```

```
  <RATE NAME="ContractApprovals" RATE_CODE="LSCM_CONTRACT_APPROVALS"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
  <RATE NAME="SubmitContract" RATE_CODE="LSCM_SUBMIT_CONTRACT"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
  <RATE NAME="ExecuteContract" RATE_CODE="LSCM_EXECUTE_CONTRACT"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="UIDCONTRACT" TYPE="I"/>
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="SECTOKEN" TYPE="S"/>
  </RATE>
  <RATE NAME="TerminateContract" RATE_CODE="LSCM_TERMINATE_CONTRACT"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
  <RATE NAME="ReviseContract" RATE_CODE="LSCM_REVISE_CONTRACT"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="CONTRACT_TYPECODE" TYPE="S"/>
  </RATE>
  <RATE NAME="CalculateTerms" RATE_CODE="LSCM_CALCULATE_TERMS"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
  <RATE NAME="ApplyProductTerm" RATE_CODE="LSCM_SHARE_ITEM_TERMS"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="ACCOUNT_ID" TYPE="S"/>
  </RATE>
  <RATE NAME="ApplyItemTerm" RATE_CODE="LSCM_SHARE_ITEM_TERMS"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="ACCOUNT_ID" TYPE="S"/>
  </RATE>
  <RATE NAME="ApplyGroupTerm" RATE_CODE="LSCM_SHARE_ITEM_TERMS"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
  <RATE NAME="Approve" RATE_CODE="LSCM_APPROVE" OPERATING_COMPANY="LODESTAR"
JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="DETAILS" TYPE="S"/>
  </RATE>
  <RATE NAME="Reject" RATE_CODE="LSCM_REJECT" OPERATING_COMPANY="LODESTAR"
JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="DETAILS" TYPE="S"/>
  </RATE>
</RATE_SCHEDULES>
```

Adding Actions

To define the rate schedule executed by custom actions, add a RATE element to the copy of the "ContractRates.xml" file.

```
<RATE_SCHEDULES>
  <RATE NAME="CUSTOM" RATE_CODE="LSCM_CUSTOM_ACTION"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
  </RATE>
</RATE_SCHEDULES>
```

Creating the Dictionary File

Dictionary files define the labels that will be visible on the user interface for the custom actions defined in the "ContractMenu.xml" file. Dictionary files also define labels from other files, including custom ASP or HTML pages you might develop.

Dictionary files must be located in the same folder as the "ContractMenu.xml" file (or other custom files).

Without proper dictionary files in place, all item names will be appear in brackets ({ }).

Note: When you create or update a "dictionary.dic" file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

"Dictionary.dic" File Structure

The basic structure of the "dictionary.dic" file, when used with contract actions, is as follows:

```
<dictionary>
  <CM>
    <DROPDOWN>
      <CUSTOM ENU="Custom Action"/>
    </DROPDOWN>
  </CM>
</dictionary>
```

where:

dictionary: is the root element of the "dictionary.dic" file.

CM: is an element which contains labels for the Contract screens.

DROPDOWN: is an element which contains labels for the Actions drop-down list.

<NAME>: is an element that corresponds to the "name" attribute of a menu section, division, or item. In the above example, "CUSTOM" is the name of the custom action. <NAME> elements have the following attributes:

- **<LANGUAGE>**: a three letter ISO code that designates the language of the element. ENU is English, and is the default. When creating dictionary files or entries for other languages, the appropriate ISO code is used. For example, for French dictionary files, you would use "FRA". For Italian dictionary files you would use "ITA".

Example

To create a label for the custom action in the above example, you would create the following in the "dictionary.dic" file.

```
<dictionary>
  <CM>
    <DROPDOWN>
      <CUSTOM ENU="Custom Action"/>
    </DROPDOWN>
  </CM>
</dictionary>
```

In this example, the custom action would be labeled "Custom Action."

Securing Actions

Securing Actions allows you to restrict which users (or groups of users) can access your custom actions.

To secure custom actions, use the following steps:

1. Add security nodes to the "ContractMenu.xml" file. These nodes define the security privileges associated with your custom actions. See **"ContractMenu.XML" File Structure - Security Nodes** on page 7-50 for details about editing this file to add security nodes.
2. Create a permissions schema file in the **C:\LODESTAR\CFG** folder. This file defines the security nodes used to restrict access to your custom tabs. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface. See **"Permission.*.XSD" File Structure** on page 7-51 for details about creating this file.
3. Create a permissions dictionary file in the **C:\LODESTAR\CFG** folder. This file defines the labels for the security nodes that will appear on the Security Administration user interface. See **Creating Permissions Dictionary Files** on page 10-11 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about creating Permissions Dictionary files.

Adding Security Nodes to the "ContractMenu.XML" File

Adding security nodes to he "ContractMenu.xml" file defines the security privileges associated with your custom actions and enables securing those custom tabs on the Energy Information Platform user interface.

"ContractMenu.XML" File Structure - Security Nodes

The structure of the "ContractMenu.xml" file changes slightly when securing actions, as follows (changes noted in bold):

```
<menubar>
  <menu name="Actions">
    <item name="CUSTOM" param="CUSTOM" prompt="Custom" action="../cm/
DropDownActions.asp" icon="save" secure="//CONTRACTTABS/ACTIONS/@CUSTOM"/>
  </menu>
</menubar>
```

where:

secure: is an attribute on an item that specifies the security nodes associated with the tab. The value specified for this attribute represents the xml structure in the Permissions schema (*.XSD) file used to define the security nodes associated with the tabs (see **Creating Permissions Schema Files** on page 7-51 for more information about permission schema files). In the above example, CUSTOM is a node of ACTIONS, which is a node of CONTRACTTABS (which is a node of PERMISSIONS). The format for this attribute is as follows:

```
//<section_node>/ACTIONS[/<action_node>]
```

where:

- <section_node> is the name of the security node associated with the tabs on the Contracts screen.

  Example: **`//CONTRACTTABS`**

- <action_node> is the name of the security node associated with the action. Action nodes are predeeded by an ampersand (@), and must be in UPPER case.

  Example: **`//CONTRACTTABS/ACTIONS/@CUSTOM`**

Example

To add security nodes to the custom tab described, you would edit the "ContractMenu.xml" file as follows (changed noted in **bold**).

```
<menubar>
  <menu name="Actions">
    <item name="CUSTOM" param="CUSTOM" prompt="Custom" action="../cm/
DropDownActions.asp" icon="save" secure="//CONTRACTTABS/ACTIONS/@CUSTOM"/>
  </menu>
</menubar>
```

Creating Permissions Schema Files

Permissions schemas are hierarchical XML structures that define the security nodes used to restrict access to your custom menu items. The nodes defined in this file appear in the Feature Privileges tree on the Privileges:Features screens (User and Group) on the Security Administration user interface.

Permissions schema files use the following naming convention:

permissions.<CUSTOM_NAME>.xsd

where:

- <CUSTOM_NAME> is a unique name that identifies the schema file.

Note: When you create or update a "permissions.*.xsd" file, you must restart Microsoft Internet Information Services (IIS) on the web server in order for the changes to take effect. You can do this from the Services Windows Control Panel (IIS Admin Service). Consult your Windows documentation or online help for more information about IIS.

"Permission.*.XSD" File Structure

The structure of the permission schema file is as follows:

```
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:attribute name="[ACTION_NAME]" type="xs:boolean" default="false">
    <xs:annotation>
      <xs:appinfo>PERM</xs:appinfo>
    </xs:annotation>
  </xs:attribute>
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="CONTRACTTABS" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="ACTIONS" minOccurs="0">
                      <xs:complexType>
                        <xs:attribute ref="[ACTION_NODE]"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

where:

ACTION_NAME: is the name of a custom action. You would include an "attrubute" element for each custom action.

PERMISSIONS: is the parent node. All security nodes are children of the PERMISSIONS node.

CONTRACT: is the node that corresponds to the Contract screen.

CONTRACTTABS: is the node that corresponds to the Contract tabs section in the "ContractTabs.xml" file.

ACTIONS: is the node that corresponds to the menu element in the "ContractMenu.xml" file.

ACTION_NODE: is a node that corresponds to the custom action in the "ContractMenu.xml" file.

How to create a permissions schema file:

To create a permissions schema file, use the following steps.

1. Create a text file in the C:\LODESTAR\CFG folder using the following naming scheme:

   permissions.<CUSTOM_NAME>.xsd

   where:

   - **<CUSTOM_NAME>** is a unique name

2. Copy the contents of the **permissions.500_datasources.xsd** file from the C:\LODESTAR\Bin folder into this new file. The contents of this file are as follows:

```
<xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="DATASOURCE" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

3. Add an "attribute" element before the PERMISSIONS node that defines the name of the custom action. This element should use the following syntax:

```
<xs:schema elementFormDefault="qualified"
  attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:attribute name="CUSTOM" type="xs:boolean" default="false">
    <xs:annotation>
      <xs:appinfo>PERM</xs:appinfo>
    </xs:annotation>
  </xs:attribute>
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="DATASOURCE" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

4. Change the DATASOURCE node to "CONTRACT." This is the parent node of the CONTRACTTABS node.

```
<xs:schema elementFormDefault="qualified"
attributeFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/
XMLSchema">
  <xs:element name="PERMISSIONS">
```

```
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5. Add a CONTRACTTABS node (or nodes) under the CONTRACT node element. This element should use the following syntax:

```
...
    <xs:element name="CONTRACT" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="CONTRACTTABS" minOccurs="0">
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
...
```

6. Add an ACTIONS node under the CONTRACTTABS node element. This element should use the following syntax:

```
...
    <xs:element name="CONTRACT" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="CONTRACTTABS" minOccurs="0">
            <xs:complexType>
              <xs:all>
                <xs:element name="ACTIONS" minOccurs="0">
                </xs:element>
              </xs:all>
            </xs:complexType>
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
...
```

7. Add an ACTION_NODE node for each custom action under the ACTIONS node element. This element should use the following syntax:

```
...
    <xs:element name="CONTRACT" minOccurs="0">
      <xs:complexType>
        <xs:all>
          <xs:element name="CONTRACTTABS" minOccurs="0">
            <xs:complexType>
              <xs:all>
                <xs:element name="ACTIONS" minOccurs="0">
                  <xs:complexType>
                    <xs:attribute ref="CUSTOM"/>
                  </xs:complexType>
                </xs:element>
              </xs:all>
            </xs:complexType>
          </xs:element>
        </xs:all>
      </xs:complexType>
    </xs:element>
...
```

Example

The permission schema file (permissions.custom.xsd) for the custom action described above would be as follows (nodes denoted in **bold**):

```
<!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Al Bresnahan
(Lodestar Corp.) -->
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:attribute name="CUSTOM" type="xs:boolean" default="false">
    <xs:annotation>
      <xs:appinfo>PERM</xs:appinfo>
    </xs:annotation>
  </xs:attribute>
  <xs:element name="PERMISSIONS">
    <xs:complexType>
      <xs:all>
        <xs:element name="CONTRACT" minOccurs="0">
          <xs:complexType>
            <xs:all>
              <xs:element name="CONTRACTTABS" minOccurs="0">
                <xs:complexType>
                  <xs:all>
                    <xs:element name="CUSTOM" minOccurs="0"/>
                    <xs:element name="ACTIONS" minOccurs="0">
                      <xs:complexType>
                        <xs:attribute ref="CUSTOM"/>
                      </xs:complexType>
                    </xs:element>
                  </xs:all>
                </xs:complexType>
              </xs:element>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Creating Permissions Dictionary Files

Just as menu items used dictionary files to define the labels that will be visible on the user interface, security nodes defined in permission schema files need dictionary files to define the labels for the security nodes that will appear on the Security Administration user interface.

Permission dictionary files must be located in the C:\LODESTAR\CFG folder.

Without proper dictionary files in place, all security nodes, including nodes for sections, divisions, and items will be appear in brackets ({ }). For instance, if you didn't create a permissions dictionary.dic file for the permissions schema described above, the privileges would look like the following in the Feature Privileges tree on the Security Administration the user interface:

Example Permissions Dictionary File - Custom Action

To create labels for the security nodes in the above example (custom action), you would create the following file ("permissions.custom.dic"):

```
<dictionary>
  <security>
    <ATTRIBUTES>
      <CUSTOM ENU="Custom Action"/>
    </ATTRIBUTES>
    <PERMISSIONS ENU="Features">
      <CONTRACT ENU="Contract Management">
        <CONTRACTTABS ENU="Contract Tab Set">
          <CUSTOM ENU="Custom Tab"/>
        </CONTRACTTABS>
      </CONTRACT>
    </PERMISSIONS>
  </security>
</dictionary>
```

See **Creating Permissions Dictionary Files** on page 10-11 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about creating Permissions Dictionary files.

Creating Custom Approval Processes

Standard contract management approval processes use the approvals functionality of the Energy Information Platform Work Queues. This approach provides considerable flexibility in the creation of approval procedures, but requires that users perform the approval actions (i.e. approving or rejecting the contract) via either the Approvals tab or the Work Queue Approvals screen. In addition, in the case of multi-step approval procedures, this approach requires that each approval step be manually approved or rejected.

However, there may be circumstances in which it may be appropriate for single approval action to complete multiple subsequent approval steps. In situations where this is required or desired, you can create custom approval processes that can conditionally perform multiple approvals within a single process. You can create custom approval processes of this type by adding custom Actions to the Actions menu of the Contracts screen (as describe in the preceding section).

This section outlines how to create custom approval processes, including:

- **Define Approval Process in Rules Language**
- **Create Custom Actions**
- **Enable the Extended Approvals Tab**

Define Approval Process in Rules Language

The first step is define the approval process (or processes) you wish to employ using Oracle Utilities Rules Language. For each approval process you wish to create, you would create appropriate records in the Rate Form and Rate Form Version tables, and then configure the Rules Language to perform the specific approval logic required. If creating multiple approval processes, any logic shared between more than one process might be best suited to be configured as a Rider.

One particular factor to consider when creating custom approval processes using Rules Language is if there are any specific conditions that impact the approval logic. For example, a custom approval process might perform certain logic based on the current External Status of the contract, or if the Margin is above a certain threshold, or if a specific term has been defined for the contract (or for one of the accounts or service points associated with the contract). Using IF THEN or SELECT statements in the Rules Language, a custom approval process can follow any number of different paths to perform approval processing.

You also need to identify the specific input parameters the approval process will require. These include:

- The Contract ID of the contract (typically specified as CONTRACT_ID)
- The Revision number of the contract (typically specified as REVISION)
- Other values used to determine the flow of business logic within the rate schedule. Examples might include:
    - Contract Category
    - Contract Type
    - Contract External Status
    - Contractee
    - Counter Party
    - Margin
    - Presence of a specific Term
    - Value of a specific Term

Example: Contractee ID Approvals - Rules Language Identifiers

For example, suppose you create an approval process that performs specific logic based on the Contractee ID of the contract. This process is defined by a rate schedule called "CONTRACTEE_ID_APPROVAL". The Rules Language for a process of this type might use the following input identifiers:

- CONTRACT_ID (the Contract ID)
- REVISION (the Contract Revision)
- CONTRACTEE_ID (the Contract's Contractee ID)

Create Custom Actions

The next step is to create add custom actions to the Actions menu for each custom approval process you defined. This requires creating copies of the "ContractMenu.xml" and "ContractRates.xml" files. See **Customizing the Actions Menu** on page 7-43 for details about creating custom actions.

Example: Contractee ID Approvals - ContractMenu.xml

The "ContractMenu.xml" files defines the custom action that will appear on the Actions menu, and the specific Contract Status Codes, External Status Codes, or Contract Categories for which the action will be available.

```
<menubar>
  <menu name="Actions">
    <item name="CONTRACTEE_ID_APPROVAL" param="CONTRACTEE_ID_APPROVAL"
action="../cm/DropDownActions.asp" icon="approval" include=" INAPPROVAL
APPROVED "/>
  </menu>
</menubar>
```

Example: Contractee ID Approvals - ContractRates.xml

The "ContractRates.xml" files defines the rate schedule to execute, as well as the specific data to be passed to the rate schedule from the contract that indicates the specific approval logic to execute, in this case, the Contractee ID.

```
<RATE_SCHEDULES>
  <RATE NAME="CONTRACTEE_ID_" RATE_CODE="CONTRACTEE_ID_APPROVAL"
OPERATING_COMPANY="LODESTAR" JURISDICTION="LODESTAR">
    <RATE_INPUT NAME="CONTRACT_ID" TYPE="S"/>
    <RATE_INPUT NAME="REVISION" TYPE="I"/>
    <RATE_INPUT NAME="CONTRACTEE_ID" TYPE="S"/>
  </RATE>
</RATE_SCHEDULES>
```

Enable the Extended Approvals Tab

The last step is to enable the Extended Approvals tab. The standard Approvals tab on the Contracts screen allows only manual approval of each step in an approval process. The Extended Approvals tab allows the use of custom approval processes.

To enable this feature, check the "Extended Approvals" feature checkbox in the Features tree in the Security Administration screen. The path to this check box is as follows:

Contract Management**->**Contract Tab Set**->**Approvals Tab**->**Extended Approvals.

See **Contract Management Features** on page 8-2 for more information about available contract management features.

Chapter 8

Configuring Oracle Utilities Quotations Management Security

This chapter describes how to configure Oracle Utilities Security to work with the Oracle Utilities Quotations Management application, including:

- **Oracle Utilities Quotations Management Security**
- **Oracle Utilities Quotations Management Approvals**

Oracle Utilities Quotations Management Security

This section describes the securable features of the Oracle Utilities Quotations Management application, including:

- **Contract Management Features**
- **Pricing Features**
- **Important Notes about Assigning Oracle Utilities Quotations Management Permissions**

Contract Management Features

- **Search Contract Type**: Enables the Contract->Contract Types menu item on the Tools and Utilities menu.
- **Search Contract Product**: Enables the Contract->Contract Products menu item on the Tools and Utilities menu.
- **Search Term**: Enables the Contract->Terms menu item on the Tools and Utilities menu.
- **Search Term Category**: Enables the Contract->Term Categories menu item on the Tools and Utilities menu.
- **Templates**: Enables the Contract->Templates menu item on the Tools and Utilities menu.
- **Term Templates**: Enables the Contract->Term Templates menu item on the Tools and Utilities menu.
- **Create Contract Type**: Enables the Add function on the Contract Types screen.
    - **Insert**: Enables inserting of contract types
- **Contract Type Tab Set** permissions include:
    - **Basics Tab**: Enables the Basics tab on the Contract Type screen.
        - **Update**: Enables the Modify function on the Basics tab of the Contract Type screen.
        - **Delete**: Enables the Delete function on the Basics tab of the Contract Type screen.
    - **Terms Tab**: Enables the Terms tab on the Contract Type screen.
        - **Insert**: Enables the Add function on the Terms tab of the Contract Type screen.
        - **Update**: Enables the modify function on the Terms tab of the Contract Type screen.
        - **Delete**: Enables the delete function on the Terms tab of the Contract Type screen.
    - **Documents Tab**: Enables the Documents tab of the Contract Type screen.
        - **Insert**: Enables the Add function on the Documents tab of the Contract Type screen.
        - **Update**: Enables the modify function on the Documents tab of the Contract Type screen.
        - **Delete**: Enables the delete function on the Documents tab of the Contract Type screen.
    - **Approvals Tab**: Enables the Approvals tab on the Contract Type screen.
        - **Update**: Enables the modify function on the Approvals tab of the Contract Type screen.

        - **Delete**: Enables the delete function on the Approvals tab of the Contract Type screen.
    - **Reports**: Enables the Reports tab on the Contract Type screen.
        - **Insert**: Enables the Add function on the Reports tab of the Contract Type screen.
        - **Update**: Enables the modify function on the Reports tab of the Contract Type screen.
        - **Delete**: Enables the delete function on the Reports tab of the Contract Type screen.
- **Create Contract Product**: Enables the Add function on the Contract Product screen.
- **Contract Product Tab Set** permissions include:
    - **Basics Tab**: Enables the Basics tab on the Contract Product screen.
        - **Update**: Enables the Modify function on the Basics tab of the Contract Product screen.
        - **Delete**: Enables the Delete function on the Basics tab of the Contract Product screen.
    - **Terms Tab**: Enables the Terms tab on the Contract Product screen.
        - **Insert**: Enables the Add function on the Terms tab of the Contract Product screen.
        - **Update**: Enables the modify function on the Terms tab of the Contract Product screen.
        - **Delete**: Enables the delete function on the Terms tab of the Contract Product screen.
- **Create Contract**: Enables the Add function on the Contracts screen.
    - **Auto Generate Contract ID**: If selected, Contract IDs are automatically generated by the application.
    - **Hide Counter Party:** If selected, the Counter Party field on the Create Contract screen is hidden from view.
- **Search Contract**: Enables the Search function on the Contracts screen.
- **Contract Tab Set** permissions include:
    - **Basics Tab**: Enables the Basics tab on the Contract screen.
    - **Items Tab**: Enables the Items tab on the Contract screen.
        - **Add Account**: Enables the Add function on the Items screen.
        - **Add Service Point**: Enables the Add Service Point function on the Items screen.
        - **Delete Items**: Enables the Delete Items function on the Items screen.
        - **Enable Groups**: Enables Groups on the Items tab of the Contract screen.
            - **Insert**: Enables the Add Groups option on the Items tab of the Contract screen.
            - **Delete**: Enables the delete function Delete Group option on the Items tab of the Contract screen.
            - **Move**: Enables the Move to Active Group option on the Items tab of the Contract screen.
    - **Terms Tab**: Enables the Terms tab on the Contract screen.
        - **Insert**: Enables the Add function on the Terms tab of the Contract screen.
        - **Delete**: Enables the delete function on the Terms tab of the Contract screen.

        - **Update**: Enables the modify function on the Terms tab of the Contract screen.
    - **Documents Tab**: Enables the Documents tab on the Contract screen.
        - **Insert**: Enables the Add function on the Documents tab on the Contract screen.
        - **Delete**: Enables the delete function on the Documents tab of the Contract screen.
        - **Update**: Enables the modify function on the Documents tab of the Contract screen.
        - **View**: Enables the View Document function on the Documents tab of the Contract screen.
    - **Approvals Tab**: Enables the Approvals tab on the Contract screen.
        - **Extended Approvals**: Enables the Extended Approvals function on the Approvals tab of the Contract screen.
    - **History Tab**: Enables the History tab on the Contract screen.
        - **Insert**: Enables the Add function on the History tab on the Contract screen.
    - **Related Contracts Tab**: Enables the Related Contracts tab on the Contract screen.
        - **Insert**: Enables the Add function on the Related Contracts tab on the Contract screen.
    - **Work Queue Detail**: Enables the Work Queue Detail function on the Related Contracts tab on the Contract screen.
    - **Actions**: Enables the Actions function on the Related Contracts tab on the Contract screen.
        - **Update**: Enables the modify function on the Contracts tab of the Contract screen.
        - **Save Contract Report**: Enables the Save Contract Report function on the Contracts tab of the Contract screen.
        - **Calculate Terms**: Enables the Calculate Terms function on the Contracts tab of the Contract screen.
        - **View Calculations**: Enables the View Calculations function on the Contracts tab of the Contract screen.
        - **View Contract**: Enables the View Contract function on the Contracts tab of the Contract screen.
        - **Ready for Approval**: Enables the Ready for Approval function on the Contracts tab of the Contract screen.
        - **Submit Contract**: Enables the Submit Contract function on the Contracts tab of the Contract screen.
        - **Execute Contract:** Enables the Execute Contract function on the Contracts tab of the Contract screen.
        - **Terminate Contract:** Enables the Terminate Contract function on the Contracts tab of the Contract screen.
        - **Revise Contract:** Enables the Revise Contract function on the Contracts tab of the Contract screen.
- **Run Reports**: Enables the Run Reports option on the Contracts menu.
- **View Reports**: Enables the View Reports option on the Contracts menu.

Pricing Features

Pricing permissions enable the menu options on the Pricing and Contracts menu. To enable the functions on the screens corresponding to these menu items, you must also enable the appropriate Contract Management, Usage, and Interval Data Manager (IDM) features. The corresponding permissions are indicated in parentheses. Pricing features include:

- **Search Contracts**: Enables the Search function on the Contracts screen (Contract Management Permissions: Search Contract).
- **Add a Forecast**: Enables the Add function on the Forecast Search screen (Usage Permissions: Add a Forecast).
    - **Insert**: Enables the Add function on the Forecast Search screen.
    - **Delete**: Enables the Delete function on the Forecast List screen.
- **Add Forecast Values**: Enables the Add function on the Forecast Input screen (Usage Permissions: Add Forecast Values).
    - **Insert**: Enables the Add function on the Forecast Input screen.
    - **Delete**: Enables the Delete function on the Forecast Input screen.
    - **Validate**: Enables the Validate function on the Forecast Input screen.
    - **Finalize**: Enables the Finalize function on the Forecast Input screen.
    - **Graph**: Enables the Graph function on the Forecast Input screen.
- **Search Forecasts**: Enables the Search function on the Forecast Search screen (Usage Permissions: Search Forecasts).
- **Validate Usage**: Enables the Validate Usage menu option on the Pricing and Contracts menu (Usage Permissions: Validate Usage).
- **Run Reports**: Enables the Run Reports option on the Pricing and Contracts menu.
- **View Reports**: Enables the View Reports option on the Pricing and Contracts menu.

Important Notes about Assigning Oracle Utilities Quotations Management Permissions

By default, the Oracle Utilities Quotations Management features restrict access to all Oracle Utilities Quotations Management functions and screens. To allow users access to Oracle Utilities Quotations Management functionality, create users and groups and enable appropriate permissions for each.

Oracle Utilities Quotations Management Approvals

The Approvals functionality of Oracle Utilities Quotations Management uses the Oracle Utilities Work Queues to filter contract approvals based on user id and approval levels (and if appropriate, other user-defined criteria).

When setting up Work Queue Types and Approval Procedures for use with Oracle Utilities Quotations Management, you must also apply the appropriate security to the Work Queue Open Items table to allow Oracle Utilities Quotations Management users to view, approve, and reject approvals associated to contracts. See **Securing Work Queue Items** on page 7-12 in the *Oracle Utilities Energy Information Platform Configuration Guide* for detailed information about securing Work Queues items.

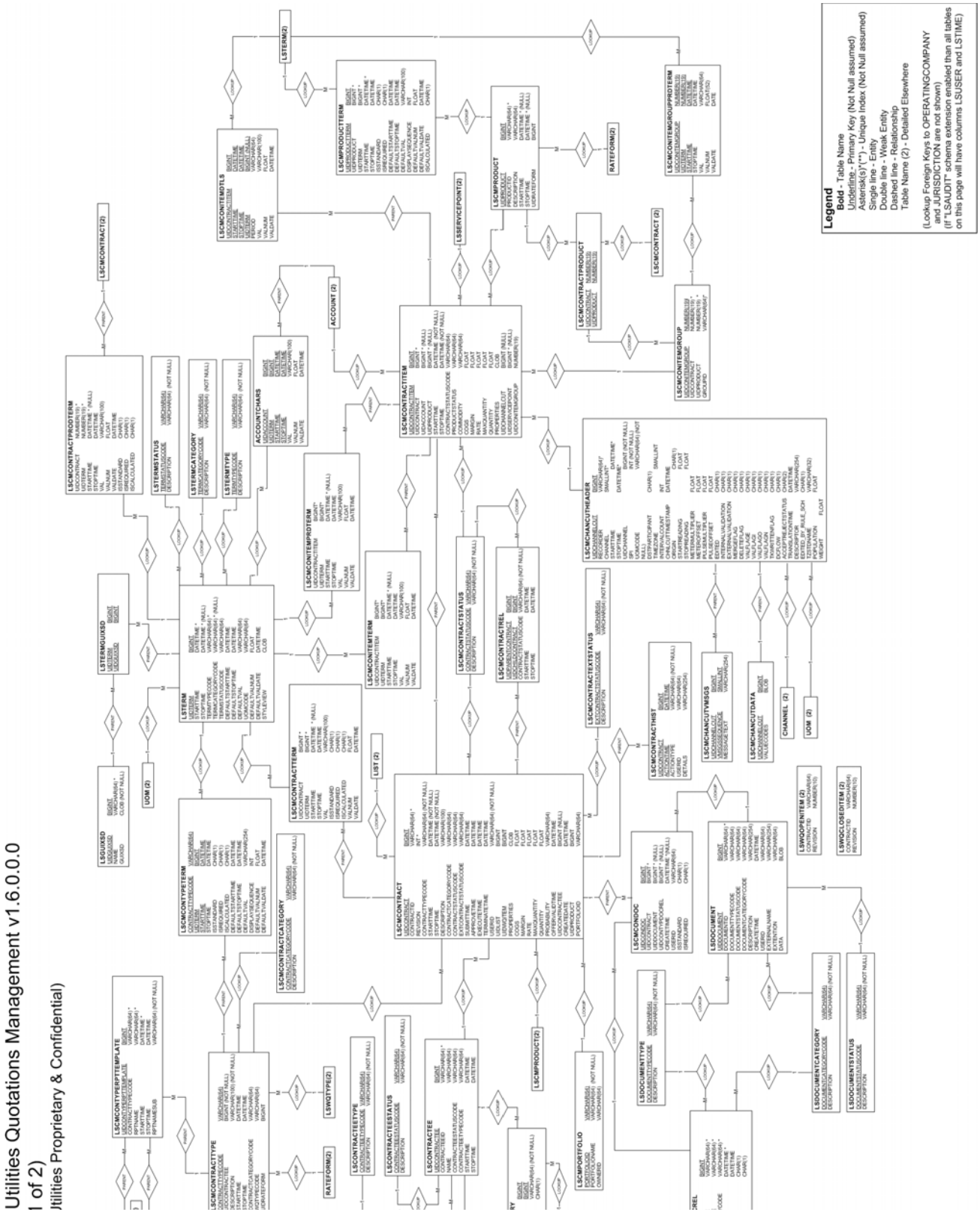
Appendix A

Oracle Utilities Data Repository Quotations Management Database Schema

This appendix includes a diagram of the Oracle Utilities Data Repository Quotations Management database schema (v1.6.1.0.0) that provides details regarding the table and columns in the Oracle Utilities Quotations Management tables, as well as the relationships between these tables in the Oracle Utilities Data Repository. This information is very useful when writing Rules Language statements or constructing database queries. This includes:

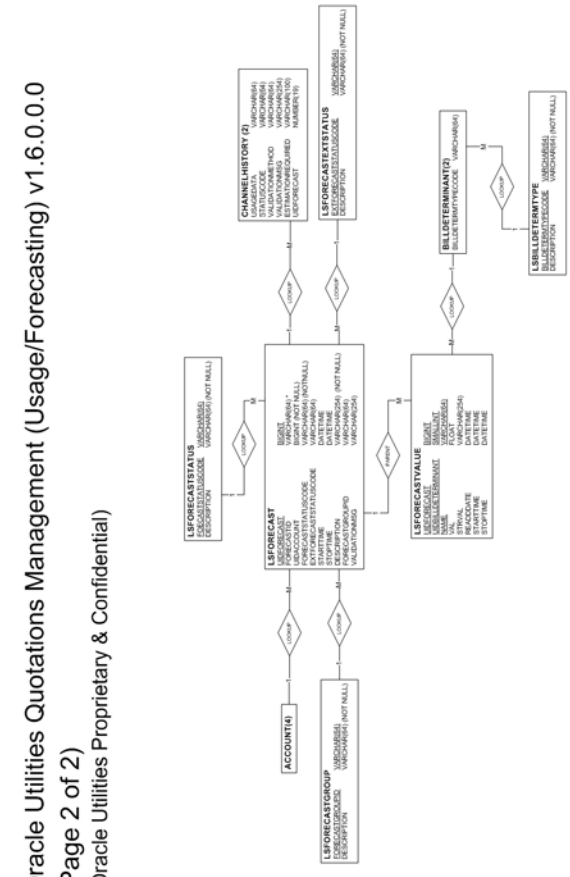
- **Oracle Utilities Quotations Management Database Schema - Contracts**
- **Oracle Utilities Quotations Management Database Schema - Usage**

Oracle Utilities Quotations Management Database Schema - Contracts



Oracle Utilities Quotations Management Database Schema - Usage

Oracle Utilities Quotations Management (Usage/Forecasting) v1.6.0.0.0
(Page 2 of 2)
(Oracle Utilities Proprietary & Confidential)



Index