Oracle Utilities Billing Component

Installation and Configuration Guide, Volume 1

Release 1.6.1.23 for Windows

E18206-24

December 2018

(Revised April 2020)

Oracle Utilities Billing Component/Billing Component Installation and Configuration Guide, Volume 1, Release 1.6.1.23 for Windows

E18206-24

Copyright © 1999, 2018 Oracle and/or its affiliates. All rights reserved.

Primary Author: Lou Prosperi

Contributor: Steve Pratt

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

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

Copyright © 1999-2000 The Apache Software Foundation. All rights reserved.

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

What’s New

New Features in the Oracle Utilities Billing Component Installation and Configuration Guide, Volume 1

This chapter outlines the new features of the 1.6.x.x release of Oracle Utitlites Billing Component 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 13-5

Chapter 1

Overview

This chapter provides an overview of the installation of Oracle Utilities Billing Component, and configuration of the billing and contract management functionality of Oracle Utilities Billing Component, including:

- **Installation and Configuration Overview**
- **What is this book?**
- **Oracle Utilities Billing Component Glossary**

Installation and Configuration Overview

Installing Oracle Utilities Billing Component, and configuring the billing and contract management functionality of Oracle Utilities Billing Component involves the following steps:

Installation

- Set up the Oracle Utilities Billing Component network environment as described in **Chapter 2**: **Oracle Utilities Billing Component Network Environment and Database Installation.**
- Install workstation and application server applications used by Oracle Utilities Billing Component on server and client machines as described in **Chapter 3**: **Installing the Oracle Utilities Billing Component Application Software** and **Chapter 3**: **Oracle Utilities Data Repository Schema Creation** of the *Oracle Utilities Energy Information Platform Installation Guide*.
- Install Oracle Utilities Billing Component web components on web server as described in **Chapter 3**: **Installing the Oracle Utilities Billing Component Application Software**.

Billing Configuration

- Set up Oracle Utilities Billing Component database records as described in **Chapter 5**: **Setting Up Oracle Utilities Billing Component Database Records**.
- Set up overrides and special events used by Oracle Utilities Billing Component as described in **Chapter 6**: **Setting Up and Applying Overrides and Special Events**.
- Set up Oracle Utilities Billing Component global defaults and settings as described in **Chapter 7**: **Setting Global Defaults and Applying Settings to Accounts, Rate Codes, and Rate Schedules**.
- Set up Oracle Utilities Billing Component to run in batch mode as described in **Chapter 8**: **Setting Up Billing to run in Batch Mode**.
- Set up and configure security for use with Oracle Utilities Billing Component web as described in **Chapter 9**: **Configuring Oracle Utilities Billing Component Security**.

Contract Management Configuration

- Set up and configure contract management database records as described in **Chapter 11**: **Setting Up Contract Management Database Tables**.
- Set up and configure contract management terms, products, and contract types as described in **Chapter 12**: **Setting Up Terms, Products, and Contract Types**.
- Add customized business logic to contract management as described in **Chapter 13**: **Adding Customized Contract Management Business Logic**.
- Set up and configure security for contract management as described in **Chapter 14**: **Configuring Contract Management Security**.

What is this book?

This book describes how to install Oracle Utilities Billing Component, and how to configure the billing and contract management functionality of Oracle Utilities Billing Component. Configuration of the receivables management and workflow management of Oracle Utilities Billing Component is described in Volume 2. This book includes the following:

- **Chapter 1**: **Overview** (this chapter) provides an overview of the Oracle Utilities Billing Component installation and configuration process
- **Chapter 2**: **Oracle Utilities Billing Component Network Environment and Database Installation** describes how to install the Oracle Utilities Billing Component database.
- **Chapter 3**: **Installing the Oracle Utilities Billing Component Application Software** describes how to install Oracle Utilities Billing Component application software on an application server, on a web server, and on client machines.
- **Chapter 4**: **Billing Rules and Definitions** describes billing rules and definitions used by Oracle Utilities Billing Component.
- **Chapter 5**: **Setting Up Oracle Utilities Billing Component Database Records** describes how to set up and configure database records used by Oracle Utilities Billing Component.
- **Chapter 6**: **Setting Up and Applying Overrides and Special Events** describes how to set up and apply overrides and special events used by Oracle Utilities Billing Component.
- **Chapter 7**: **Setting Global Defaults and Applying Settings to Accounts, Rate Codes, and Rate Schedules** describes how to set up global defaults and settings used by Oracle Utilities Billing Component.
- **Chapter 8**: **Setting Up Billing to run in Batch Mode** describes how to set up Oracle Utilities Billing Component to run in batch mode.
- **Chapter 9**: **Configuring Oracle Utilities Billing Component Security** describes how to configure security for use with the web-based Oracle Utilities Billing Component application.
- **Chapter 10**: **Contract Management Rules Language Rate Schedules, Riders, and Lists** describes the Oracle Utilities Rules Language rate schedules, riders, and lists used by the contract management functionality of Oracle Utilities Billing Component.
- **Chapter 11**: **Setting Up Contract Management Database Tables**. describes how to set up database records used by the contract management functionality of Oracle Utilities Billing Component.
- **Chapter 12**: **Setting Up Terms, Products, and Contract Types** describes how to set up and configure terms, products, and contract types used by the contract management functionality of Oracle Utilities Billing Component.
- **Chapter 13**: **Adding Customized Contract Management Business Logic** describes how to add customized business logic to contract management functionality of Oracle Utilities Billing Component.
- **Chapter 14**: **Configuring Contract Management Security** describes how to configure security for use with contract management functionality of Oracle Utilities Billing Component.
- **Appendix A**: **Creating a CIS Transaction Record Output File** describes how to create a CIS transaction output files, used when passing billing calculations and results to a Customer Information System (CIS).
- **Appendix B**: **Mapping Rate Codes Between Rate Schedules** describes how to map rate codes to multiple rate schedules.

- **Appendix C**: **Oracle Utilities Data Repository - Contract Management Database Schema** provides a database schema diagram of the tables in the Oracle Utilities Data Repository used by the contract management functionality of Oracle Utilities Billing Component.

Oracle Utilities Billing Component Glossary

This section provides a glossary of terms used in Oracle Utilities Billing Component and other Oracle Utilities product documentation.

Term	Definition
Available	An account is "available" for billing if it is "eligible" and if all data required to compute its bill is available to Oracle Utilities Billing Component.
Bill Determinant	A value used to compute a customer bill (e.g., kWh). Also referred to as *billing determinant.*
Billing Mode Flag	A value in the Oracle Utilities Data Repository -- assigned at the account-, rate code-, or rate-schedule level -- that determines how an account is billed (Approval Required, Fully Automatic, or Manual Start).
Bill Month	Revenue month; the accounting month that the income from a bill is assigned to.
Channel	An element on an interval data recorder assigned to measure a specific end use or other quantity of interest at a specific unit of measure.
Channel Group	Any user-specified combination of interval data channels whose data you want to total for billing or other purposes.
CIS Account	(*Note: your company may or may not use CIS accounts.*) In some Customer Information Systems, all billing entities are identified by an account ID—for example, customers, accounts, account service locations, or others. Oracle Utilities Billing Component, on the other hand, recognizes these entities as different levels of the billing hierarchy, and assigns them different ids accordingly. This makes it possible for Oracle Utilities Billing Component to calculate usage and charges for flexible grouping of entities. In Oracle Utilities Billing Component, the CIS Account Table is an optional table that lets you "map" entities recognized by your CIS system to entities recognized by Oracle Utilities Billing Component.
Contract	A rate form that contains special Rules Language statement specific to an account.
Cut	A data record containing interval values for one recorder channel, for the period between read dates.
Eligible	An account is "eligible" for billing if today's date falls within a user-defined number of days before and after the account's scheduled meter read date.
Group	*See Channel Group.*
IPH	Intervals per hour; reciprocal of minutes per interval (MPI).
Interval Data	A series of values that represents a customer' demand (kW) or other quantity measured on a periodic basis, such as every 5, 15, 30, or 60 minutes. Also referred to as *load data* or *time-series data.*
MPI	Minutes per interval; reciprocal of intervals per hour (IPH).

Term	Definition
Override	A modification to the tariffs normally used to compute an account's bill. An override might be long-term, such as an “economic development discount,” or it might be short-term, coinciding with a “special event” such as an interruption or maintenance period.
Post-window	Number of days after an account's scheduled read date that Oracle Utilities Billing Component's Automatic and Approval Required modules continue scanning for the account's billing data in order to bill it.
Pre-window	Number of days before an account's scheduled read date that Oracle Utilities Billing Component's Automatic and Approval Required modules begin scanning for the accounts billing data in order to bill it.
Rate Form	The generic term applied to a contract, rate schedule, or rider.
Rate Schedule	The rate form that Oracle Utilities Billing Component uses to compute the bill for an account; may include contracts and/or riders.
Read Date	Date that an account's meter is scheduled to be read. It is determined by the billing cycle code assigned to the account. Oracle Utilities Billing Component stores the read date in the Billing History record, to associate the bill with the correct billing cycle.
Recorder	A device used to capture and store interval data. The data is retrieved either manually or by telecommunications means.
Rider	A set of Rules Language statements that can be included in another rate form, avoiding the necessity of having to re-create often-used sets of statements.
Special Events	An event which causes alternative charges (overrides) to be applied to an account's bill. In Oracle Utilities Billing Component, the terms *special events* and *overrides* are used somewhat interchangeably.
Tariff Rider	A tariff rider that a billing analyst or other user has explicitly attached to an account via the Account Rider History Table.
UOM	Unit of Measure.

Part One

Installation

Part One describes installation of Oracle Utilities Billing Component, and contains the following chapters:

- **Chapter 2**: **Oracle Utilities Billing Component Network Environment and Database Installation**
- **Chapter 3**: **Installing the Oracle Utilities Billing Component Application Software**

Chapter 2

Oracle Utilities Billing Component Network Environment and Database Installation

This chapter describes the Oracle Utilities Billing Component network environment, including:

- **The Oracle Utilities Billing Component Installation Package**
- **Oracle Utilities Billing Component Network Environment**
- **Oracle Utilities Billing Component Database Installation**

The Oracle Utilities Billing Component 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 Billing Component.

Product	Installation File Name
Oracle Utilities Energy Information Platform	1.6.1.xx.0.EIP.zip
Oracle Utilities Billing Component	1.6.1.xx.0.BC.zip

The Oracle Utilities Billing Component installation package contains the following folders:

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

  See **Oracle Utilities Billing Component Database Installation** on page 2-9 for more information about installing and upgrading the Oracle Utilities Billing Component database schema.
- **Documentation**: Contains documentation for Oracle Utilities Billing Component, including:
    - Oracle Utilities Billing Component Installation and Configuration Guide, Volume 1
    - Oracle Utilities Billing Component Installation and Configuration Guide, Volume 2
    - Oracle Utilities Billing Component User's Guide (web)
    - Oracle Utilities Billing Component User's Guide (c/s)

    Note: Oracle Utilities Billing Component documentation can also be downloaded separately.

Oracle Utilities Billing Component 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 Billing Component Database Installation

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

- **Installation Requirements**
- **Installing the Database**
- **Upgrading the Billing Component 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 Billing Component 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 Billing Component application.
- The **addCE.cmd** file. This database script adds the contract management database tables and data used by Oracle Utilities Billing Component to the Oracle Utilities Data Repository.
- The **updateCE.cmd** file. This database script updates the contract management tables and data used by Oracle Utilities Billing Component in the Oracle Utilities Data Repository. See **Upgrading the Billing Component Database Schema** on page 2-11 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 **addFME.cmd** file. This database script adds the Oracle Utilities Receivables Component database tables and data to the Oracle Utilities Data Repository.
- The **BIPFME.cmd** file. This database script changes the pre-defined receivables component 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 **addWKFLW.cmd** file. This database script adds the Oracle Utilities Workflow Management database tables and data to the Oracle Utilities Data Repository.

Installing the Database

Installing the Oracle Utilities Billing Component database involves installing the Oracle Utilities Billing Component - Contract Management database schema and the Receivables Component 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.
<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.

Oracle Utilities Receivables Component

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

```
addFME [-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 database objects. This parameter is optional. If not specified, the default 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 Receivables Component 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.

Oracle Utilities Workflow Management

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

```
addWKFLW [-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 database objects. This parameter is optional. If not specified, the default 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 WorkFlow 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.

Upgrading the Billing Component Database Schema

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

- The **toFME16129.cmd** file. This database script updates the receivables management tables and data used by Oracle Utilities Billing Component 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 (Receivables Management)	v1.6.1.2.0

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

Upgrading Oracle Utilities Billing Component - Contract Management Schema

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

```
toFME16120 [-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 Billing Component 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.
- The **BIPFME.cmd** file. This script changes the pre-defined receivables component reports from Crystal Reports to Oracle Business Intelligence Publisher.

Oracle Utilities Billing Component - Contract 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.

Oracle Utilities Receivables Component Reports

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

```
BIPFME [-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 database objects. This parameter is optional. If not specified, the default user PWRLINE will own database objects.
<owner password>	The chosen password for the PWRLINE schema owner.

Verifying the Database

Verifying the Oracle Utilities Billing Component database involves verifying the billing tables (installed as part of the Oracle Utilities Energy Information Platform database schema), the Oracle Utilities Billing Component - Contract Management database schema, the Oracle Utilities Receivables Component database schema, and the Workflow Management database schema in the Oracle Utilities Date Repository.

Verification - Billing Tables

To verify that the Oracle Utilities Billing Component schema 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:
   - ACCOUNTHISTORY (Account History)
   - ACCTBILLDETUSE (Bill Determinant Use)
   - ACCTNAMEOVERHIST (Name Override History)
   - ACCTOVERRIDEHIST (Override History)
   - ACCTOVERRIDEREADY (Override Ready)
   - ACCTOVERRIDEUSE (Override Use)
   - ACCTRATECODEHIST (Rate Code History)
   - ACCTRIDERHIST (Rider History)
   - BILLCODE (Bill Code)
   - BILLDETERMINANT (Bill Determinant)
   - BILLHISTORY (Bill History)
   - BILLHISTORYEDIT (Bill History Edit)
   - BILLHISTORYEDITVALUE (Bill History Edit Value)
   - BILLHISTORYPROBLEM (Bill History Problem)
   - BILLHISTORYVALUE (Bill History Value)
   - BILLINGCYCLE (Billing Cycle)
   - BILLINGCYCLEDATE (Billing Cycle Dates)
   - RATECODE (Rate Code)

Appendix A: **Oracle Utilities Data Repository Database Schema** includes a diagram of the Oracle Utilities Data Repository database schema that provides details regarding the table and columns in the Oracle Utilities Energy Information Platform schema, as well as the relationships between these tables in the Oracle Utilities Data 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:
   - LSCMCHANCUTDATA (CM Channel Cut Data)
   - LSCMCHANCUTHEADER (CM Channel Cut Header)

- LSCMCHANCUTVMSGS (CM Channel Cut Validation Messages)
- LSCMCONTRACT (Contract)
- LSCMCONTRACTCATEGORY (Contract Category)
- LSCMCONDOC (Contract Document)
- LSCMCONTRACTEXTSTATUS (Contract External Status)
- LSCMCONTRACTHIST (Contract History)
- LSCMCONTRACTITEM (Contract Item)
- LSCMCONITEMDTLS (Contract Item Details)
- LSCMCONITEMGROUP (Contract Item Group)
- LSCMCONITEMGROUPPRDTERM (Contract Item Group Product Term)
- LSCMCONITEMPRDTERM (Contract Item Product Term)
- LSCMCONITEMTERM (Contract Item Term)
- LSCMPORTFOLIO (Contract Portfolio)
- LSCMCONTRACTREL (Contract Relationship)
- LSCMCONTRACTSTATUS (Contract Status)
- LSCMCONTRACTTERM (Contract Term)
- LSCMCONTRACTTYPE (Contract Type)
- LSCMCONTYPDOCREL (Contract Type Doc Rel)
- LSCMCONTYPERPTTEMPLATE (Contract Type Report Template)
- LSCMCONTYPETERM (Contract Type Term)
- LSCMCONTRACTEE (Contractee)
- LSCMCONTDIRECTORY (Contractee Directory)
- LSCMCONTRACTEESTATUS (Contractee Status)
- LSCMCONTRACTEETYPE (Contractee Type)
- LSDOCUMENT (Document)
- LSDOCUMENTCATEGORY (Document Category)
- LSDOCUMENTSTATUS (Document Status)
- LSDOCUMENTTYPE (Document Type)
- LSCMPRODUCT (Product)
- LSCMPRODUCTTERM (Product Term)
- LSGUIXSD (Template)
- LSTERM (Term)
- LSTERMCATEGORY (Term Category)
- LSTERMSTATUS (Term Status)
- LSTERMGUIXSD (Term Template)
- LSTERMTYPE (Term Type)

Appendix C: Oracle Utilities Data Repository - Contract Management Database Schema includes a diagram of the Oracle Utilities Billing Component Contract Management database schema (v1.6.1.0.0) that provides details regarding the table and columns in the Contract

Management schema, as well as the relationships between these tables in the Oracle Utilities Data Repository.

Verification - Receivables Component

To verify that the Receivables Component tables were installed successfully, 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:
    - ACCOUNTFME (Account FME)
    - APPLICATIONPRIORITY (Application Priority)
    - ASSISTPROGRAM (Assistance Program)
    - AUTOPAYMENT (Automatic Payment)
    - AUTOPAYPLAN (AutoPayment Plan)
    - BATCHPAYMENT (Batch Payment)
    - BATCHTRANSACTION (Batch Transaction)
    - BUDGETPLAN (Budget Plan)
    - BUDGETTYPE (Budget Type)
    - CANCELREASON (Cancel Reason)
    - CHARGETYPE (Charge Type)
    - CONTROLMSG (Control Message)
    - CONTROLMSGDATA (Control Message Data)
    - COSTCENTER (Cost Center)
    - COSTCENTERTXLATION (Cost Center Translation)
    - CREDITAPPLICATION (Credit Application)
    - DEPOSIT (Deposit)
    - INPUTMSG (Input Message)
    - INPUTMSGDATA (Input Message Data)
    - INSTALLMENTPLAN (Installment Plan)
    - JOURNALACCOUNT (Journal Account)
    - JOURNALTRANSACTION (Journal Transaction)
    - JOURNALTRANSLATION (Journal Translation)
    - LSTRANSACTION (Transaction)
    - OUTPUTMSG (Output Message)
    - OUTPUTMSGDATA (Output Message Data)
    - PAYMENT (Payment)
    - PAYMENTMETHOD (Payment Method)
    - PAYMENTSOURCE (Payment Source)
    - RECEIVABLETYPE (Receivable Type)
    - SUBLEDGERACCOUNT (SubLedger Account)

- SERVICEPLAN (Service Plan)
- TAXRECORD (Tax Record)
- TRANSACTIONID (Transaction ID)
- TRANSACTIONTYPE (Transaction Type)
- WORKQUEUEMSG (Work Queue Message)
- WORKQUEUEMSGDATA (Work Queue Message Data)
- WRITEOFFREASON (Writeoff Reason)

FME Reports Tables

- REPORT (Report)
- REPORTLIST (Report List)

Collection Tables

- ASSISTAGENCY (Assistance Agency)
- ASSISTPGMXRCVTYPE (Assistance Program X Receivable Type)
- ASSISTPLAN (Assistance Plan)
- ASSISTPLEDGE (Assistance Pledge)
- ASSISTPROGRAM (Assistance Program)
- CHECKINGACCOUNT (Checking Account)
- COLAGENCY (Collection Agency)
- COLAGENCYMSG (Collection Agency Program)
- COAGENCYPROGRAM (Collection Agency Program)
- COLAGENCYXDIRECTORY (Collection Agency X Directory)
- COLARRANGEMENT (Collection Arrangement)
- COLARRANGETYPE (Collection Arrangement Type)
- COLARRPAYMENT (Collection Arrangement Payment)
- COLEXEMPTYPE (Collection Exemption Type)
- COLEXEMPTION (Collection Exemption)
- COLPROGTYPE (Collection Program Type)
- COLPROGXACCOUNT (Collection Program X Account)
- COMOBJECT (COM Object)
- COLOBJECTHISTORY (COM Object History)
- COMOBJTYPE (COM Object Type)
- CONTEXTVALUE (Context Value)
- CREDITSCOREREPORT (Credit Score Report)
- CREDITSCOREHISTORY (Credit Score History)
- DYNAMICMSG (Dynamic Message)
- DYNAMICMSGVALUE (Dynamic Message Value)
- LETTERMSG (Letter Message)
- LSREFUND (Refund)

- PHONEMSG (Phone Message)
- PROCCONTEXTVALUE (Process Context Value)
- REFUNDREASON (Refund Reason)
- SHUTOFFMSG (Shut Off Message)
- SPCLHANDLECODE (Special Handling Code)
- VARSOURCE (Variable Source)

Verification - Workflow Management

To verify that the Workflow Management tables were installed successfully, 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:

   WorkFlow Tables

   - ACTIVITYARCHIVE (Activity Archive)
   - ACTIVITYEVENT (Activity Event)
   - ACTIVITYIMPL (Activity Implementation)
   - ACTIVITYINSTANCE (Activity Instance)
   - EVENTTYPE (Event Type)
   - PROCESS (Process)
   - PROCESSACTIVITY (Process Activity)
   - PROCESSARCHIVE (Process Archive)
   - PROCESSINSTANCE (Process Instance)
   - PROCESSVERSION (Process Version)
   - WFWQMSG (WorkFlow Work Queue Message)

Appendix A: **Oracle Utilities Data Repository Receivables Component Database Schema** and **Appendix B**: **Oracle Utilities Data Repository Workflow Management Database Schema** in the *Oracle Utilities Billing Component Installation and Configuration Guide, Volume 2* includes diagrams of the Oracle Utilities Receivables Component and Workflow Management database schemas (v1.6.1.0.0) that provide details regarding the table and columns in the Receivables Component and Workflow Management schemas, 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

Verification - Receivables Component Reports

To verify that the receivables component 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
colActivitySum	Collection Activity Summary	BIPublisher
colExemptionSum	Collection Exemptions Summary	BIPublisher
WQSum	Work Queue Summary	BIPublisher
processStatus	Process Status	BIPublisher
performanceScore	Performance Rating Summary	BIPublisher

Chapter 3

Installing the Oracle Utilities Billing Component Application Software

This chapter describes how you install the Oracle Utilities Billing Component application software, including:

- **Installing the Oracle Utilities Billing Component Software**
- **Setting Up Configuration Files on the Web Server**
- **Setting Up Configuration Files on Workstations**

Installing the Oracle Utilities Billing Component Software

This section describes how to install the Oracle Utilities Billing Component 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 Billing Component 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 (01.06.00.xx.00.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 from the Oracle Utilities Billing Component installation package (1.6.1.xx.0.BC.zip). This folder contains the following files:
    - Oracle Utilities BC 1.6.1.xx.0.msi
3. Move the "Oracle Utilities BC 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 BC 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 Billing Component software after the Oracle Utilities Energy Information Platform software has been installed, use the following procedure:

1. Navigate to the Install folder created from the Oracle Utilities Billing Component installation package (1.6.1.xx.0.BC.zip). This folder contains the following files:
    - Oracle Utilities BC 1.6.1.xx.0.msi
2. Double-click the Oracle Utilities BC 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 Workstations

You also need to set up the configuration files needed by the workstation applications such as Data Manager.

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 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.

Part Two

Billing Configuration

Part Two describes configuration of the billing functionality of Oracle Utilities Billing Component, and contains the following chapters:

- **Chapter 4**: **Billing Rules and Definitions**
- **Chapter 5**: **Setting Up Oracle Utilities Billing Component Database Records**
- **Chapter 6**: **Setting Up and Applying Overrides and Special Events**
- **Chapter 7**: **Setting Global Defaults and Applying Settings to Accounts, Rate Codes, and Rate Schedules**
- **Chapter 8**: **Setting Up Billing to run in Batch Mode**
- **Chapter 9**: **Configuring Oracle Utilities Billing Component Security**

Chapter 4

Billing Rules and Definitions

Before you can process bills for accounts and customers, you first need to understand the billing rules; that is, the rules by which bills are processed by Oracle Utilities Billing Component. This chapter describes these rules, including the following:

- **Billing Definitions**
- **General Billing Rules**
- **Selecting Accounts to Bill**
- **The Billing Process**
- **Billing Account Note/Work Queue Item Generation**
- **Computing the Bill Start and Bill Stop**
- **Summary Bill Calculation**
- **Saving the Bill History Record**
- **Saving Data from a Rate Schedule**
- **Multiple Bill History Records per Bill Period**
- **Invoice Numbering**
-

Billing Definitions

This section contains a short glossary of terms used in later sections of this chapter.

In the definitions that follow, database tables and columns are notated using the TABLE.COLUMN format, where TABLE denotes the specific database table, and COLUMN denotes the specific column in the listed database. For instance, ACCOUNT.ACCOUNTID means the Account ID column in the Account Table.

Account Stop Date: The date and time the account was terminated, located in the Stop Time column of the Account Table (ACCOUNT.STOPTIME).

Read Date: The Scheduled Read Date, located in the Read Date column of the Billing Cycle Dates Table (BILLINGCYCLEDATE.READDATE). In the following, "read date" always refers to the scheduled read date unless modified by the word "actual" (as in "actual read date"). The Read Date is:

1. The read date in the last unbilled Bill History record in which the value for Bill Time is NULL
2. The first read date after the read date in the last billed Bill History record
3. The first read date after the value in the Start Time column in the Account Table (ACCOUNT.STARTTIME).

   Note: When used with Oracle Utilities Billing Component, the READDATE value in the BILLHISTORY Table cannot be NULL.

Bill Start Time: The Start Date and Time for the bill period. Usually the same as the previous bill period's stop time plus one second. This is stored in the Start Time column of the Bill History Table (BILLHISTORY.STARTTIME).

The Bill Start Time may also be referred to as the Bill Period Start Date or Bill Period Start Time.

Bill Stop Time: The Actual Read Date, stored in the Stop Time column of the Bill History Table (BILLHISTORY.STOPTIME), usually used to compute demand determinants. It is computed as:

1. If User-Specified (only in Current/Final Bill), use it, else
2. If no interval data, use dates input along with scalar determinants, else
3. If one channel for the account, its stop time, else
4. The earliest (synchronized) stop time of multiple cuts.

If bill stop was calculated and the ACCOUNT.STOPTIME is set and less than the bill stop, use the ACCOUNT.STOPTIME as the bill stop.

The Bill Stop Time may also be referred to as the Bill End Time, Bill Period Stop Date, or Bill Period Stop Time.

If any of the Multiple Bill History Records per Bill Period options in the CIS Billing Options is selected, the Bill History start and stop times are used as is for the Bill Start and Stop Times, they are not computed. Bill History records are required in this case.

Scalar determinants are input via the Bill History (BILLHISTORY) Table, in a record with the corresponding scheduled read date. This record also contains Bill Start and Stop Times. If there is also interval data, the interval data dates override those in the Bill History record, unless the User Specified Stop (USERSPECIFIEDSTOP) column is used.

Bill Time: The date and time Oracle Utilities Billing Component computes the bill and sends results to CIS, located in the Bill Time column of the Bill History Table (BILLHISTORY.BILLTIME).

Invoice Date: The date CIS prints the bill. It is not stored in the database.

The interval data cuts used to compute energy are all those whose start time is greater than or equal to the Bill Start Time and whose start time less than the Bill Stop Time.

The interval data cuts used to compute demand are all those whose start time or stop time is between the Bill Start Time and the Bill Stop Time, but include only the data between these two Bill times. For Full Day bills, the demand is computed based on data from the midnight before the Bill Start Time to the midnight before the Bill Stop Time.

General Billing Rules

When an account is billed several times at once; for example, a weekly account in a summary bill or an account with multiple bill history records in a bill period, the bills are computed from first to last. Any saved determinants in a bill are temporarily written back to the database if the bill completes successfully, and can then be used in the next bill. They are "rolled back" (removed) from the database after the last bill for the account is processed, and are then written to the database when the account's bills are approved.

Account History Record Selection

During the preparation for billing, a record from the Account History (ACCOUNTHISTORY) Table is read for each account. The record contains information such as the Full Day Bill flag, the Contract, and the Governing Cycle. The record read is the latest one before the expected Bill Stop. The values from this record are used, so it does not matter if there is a previous Account History record and the Governing Cycle date is before the one read, the information in the one read is used. The simplest Effective Date on an Account History record is immediately after the previous bill's Read Date.

Rejections and Exceptions

A billing exception occurs when an account is eligible but not available. A rejection occurs when an available account fails in the actual bill calculation (usually due to validation errors). In addition, a rejection may occur when a user reviews a bill and "rejects" it. Reasons for rejections and exceptions are written to the Account Note (ACCOUNTNOTE) Table. The Review Billing Exceptions report see **Chapter 8**: **Review Billing Exceptions** in the *Oracle Utilities Billing Component User's Guide* or **Billing Exceptions** on page 10-5 in the *Oracle Utilities Billing Component Web User's Guide*) can be used to view billing exceptions and rejections, as well as other information in the Account Note Table.

Effective and Governing Dates

The Effective Date determines the rate components used when processing a bill, including Rate Form, factor value and season. This date can also be referenced in the Rules Language using the EFFECTIVE_DATE identifier.

The Effective Date option in the Default Billing Options dialog allows users to choose the billing date to use as the Effective Date for calculating bills. The drop-down list contains the following options: Bill Stop, Bill Start, 1st of Bill Month, Scheduled Read Date and, possibly, and Governing Date. The Governing Date option appears only if there is a Governing Cycle column in the Account History table). The default is Bill Stop. The selection, when saved, applies to all Oracle Utilities Billing Component users and all Oracle Utilities Billing Component instances started after it is saved.

Governing Cycle Date

The scheduled read date determines when an account is billed, the governing cycle date can be used to determine what rate components to use. The Governing Cycle column on the Account History table refers to the billing cycle that determines the governing date. The governing date is only determined if the Effective Date option (described previously) is set to **Governing Date**. If an account's Governing Cycle is not NULL, the governing date is the Read Date value in the Billing Cycle Date record related to Governing Cycle with the same Bill Month as the scheduled read date. If the column is NULL or there is no match, the governing date is set to the scheduled read date. If there is no scheduled read date (which could happen in a Trial Bill) then midnight of the Bill Stop is used as both the scheduled read date and the governing date.

Selecting Rate Schedule and Included Rate Forms

In all Oracle Utilities Billing Component billing modules, including Automatic Billing, Approval Required, Current/Final Bill, Bill Correction and Trial Calculation, there is a date used to retrieve the rate schedule and its included riders that are in effect on the date.

Note: In version 2.02.210 and before this is always the bill stop date. After version 2.02.210, the date will be the appropriate effective date, depending on the Effective Date Billing Option.

Selecting Factor Values

Factor values are selected based factor key code and Bill Start, Bill Stop and Effective Dates. The Prorate column in the Factor Value table determines if, and how, the factor is prorated as follows:

- **Yes**: prorate the factor over entire bill period.
- **No**: do not prorate the factor and use the value on bill stop date.
- **Effective Date**: use the factor value in effect on the Effective Date.

Selecting Season

The SELECT BILL_PERIOD Rules Language statement uses one of five dates to determine the season of the bill period: the Bill Stop date, the Bill Start date, the first day of the Bill Month, the scheduled Read Date or the Governing Date. This selection can be set by assigning a value to the BILL_PERIOD_SELECT identifier (0, 1, 2, 3 and 4, respectively). The default is whichever is the effective date, according to the Effective Date Billing Option (described previously).

Note: If the Governing Cycle column is NULL and the user assigns 4 (Governing Date) in the Rules Language, the Read Date is used. If the Read Date is also NULL, midnight of the Bill Stop is used.

Bill On Read Date

Some users specify a bill date. The two billing modes that find accounts to bill, Automatic Billing and Approval Required, must bill these accounts on that date. To support this the Account History table has a Bill on Read Date column. If this column is not NULL and equal to “Yes”, the account is billed on its scheduled Read Date, providing the date input to Automatic Billing or Approval Required is the same as the scheduled Read Date. The account will attempt to bill at the first run of these billing modes on the Read Date, and continue to try to bill all day long. **If data is not available by the end of the day, the account will not bill.** This effectively sets the account's Post Window to zero, however the Pre Window is still used as normal.

If the Bill on Read Date column is set to “Bill Stop,” the account is billed on its scheduled Read Date, providing the date input to Automatic Billing or Approval Required is the same as the scheduled Read Date, and the Bill Stop column for the Bill History record is adjusted to the read date.

Selecting Accounts to Bill

This section describes how accounts are selected to bill. An account flag indicates whether the account is billed only when the billing analyst requests it, or if its billing cycle date is to be used. In the former case, the bill can only be computed via Current/Final Bill. In the latter case, the bill can be computed via either Approval Required or Automatic Billing.

Billing Mode Flags

A Billing Mode flag can be specified in three tables, the Account (ACCOUNT) Table, the Rate Code (RATECODE) Table and the Rate Form (RATEFORM) Table. If not set to NULL or DEFAULT, the value in the Account Table is used, then the value in the Rate Code Table, and finally the value in the Rate Form Table. If all of these are set to NULL or DEFAULT, Default Billing Mode set in the CIS Billing Options is used.

Full Day Billing Rules

A Full Day Bill flag can be specified in four tables, the Account History (ACCOUNTHISTORY) Table, the Account (ACCOUNT) Table, the Rate Code (RATECODE) Table, and the Rate Form (RATEFORM) Table. If not set to NULL, the value in the Account History Table is used, then the value in the Account Table, then the Rate Code Table, and finally the value in the Rate Form Table. If all of these are set to NULL, Full Day Billing is off. If it is set to Yes or No in the Account History Table, it applies to the Bill Start and Stop dates on or after the value in the Effective Date column in the Account History Table (ACCOUNTHISTORY.STARTTIME).

Automatic and Approval Required Accounts

Approval Required Accounts and Automatic Accounts differ in one important way. When an Automatic Account is billed, its billing information is automatically approved and sent to CIS. On the other hand, data for Approval Required Accounts is sent only when a user approves the bill. This occurs when the user views the results of a bill calculation and either approves it or rejects it. Manual review is required for a bill if:

1. An account-specific flag indicates that all bills for the account require review
2. A rate code-specific flag indicates that all bills computed using the rate code require review
3. A rate schedule-specific flag indicates that all bills computed using the rate schedule require review
4. A global option indicates that all bills require review.

The account, rate code, and rate schedule flags may be ON, OFF, or NULL. If an account flag is ON or OFF, it overrides the rate code, rate schedule, and global flag. If it is NULL, the rate code, rate schedule, or global flag is used. Similarly, the rate code flag overrides the rate schedule flag, which in turn overrides the global flag.

Approval Required Accounts and Automatic Accounts share in how an account is picked for billing. The main questions asked in this determination are:

1. Is it eligible for billing (is the account's read date "near now")?
2. Is it available for billing (is all the data needed to compute the bill available, right now)?

Automatic Billing

When the Automatic Billing module (see **Chapter 5**: **Automatic Billing** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 6**: **Automatic Billing** in the *Oracle Utilities Billing Component Web User's Guide*) is running, Oracle Utilities Billing Component continuously scans for accounts that are flagged for automatic billing are eligible for billing, have all their billing data, have no previous billing errors for the current bill period, and have not yet been billed. When it finds them it performs the bill calculation, approves the bill, and sends the results to CIS.

Approval Required Billing

The Approval Required module (see **Chapter 6**: **Approval Required Billing** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 5**: **Approval Required Billing** in the *Oracle Utilities Billing Component Web User's Guide*) performs the same basic task as the Automatic Billing module, except that it bills only accounts selected by the user that are flagged for Approval Required. The Approval Required module checks for eligibility and availability for each account only once per run. In addition, each bill processed by the Approval Required module must be approved before the results are sent to CIS.

The Billing Process

This section describes the billing process, including the determination of account eligibility and availability.

Account Eligibility

An account is "eligible" for billing if the Stop Date column in the Account Table (ACCOUNT.STOPTIME) is NULL or set to a date later than 35 days from now (the current date), and one of its read dates is "near" now. If the Stop Date is not NULL, the account is billed until there is a Bill History record with a Stop Time greater than or equal to the account Stop Time.

Read dates are determined from the billing cycle code. A billing cycle code is related to each account through the Account History Table. Associated with each billing cycle code is a set of read dates and bill months. The read date closest to now is checked to see if it is close enough for billing.

An initial list of accounts that are close enough is determined by checking the Default Pre and Post Windows in the CIS Billing Options around now. The CIS Billing Options default pre- and post-windows must be greater than any corresponding values for any account. The list is then shrunk based on the following rules.

1. If a pre- or post-window is in the current Account record, that is used.

   Otherwise:

2. If a pre- or post-window is NULL in the Account record, a list of all recorders for the account is retrieved, and the Pre- and Post-window values are retrieved. The pre-window to use is the smallest one; the post-window is the largest one.
3. If the account does not have any recorders, the global default values set in the CIS Billing Options are used.

If the current date is greater than or equal to the Read Date minus the Pre-window, and less than or equal to the Read Date plus the Post-window, the bill month associated with the Read Date is retrieved.

> **Note:** If the Read Date falls within the account's Pre- and Post-windows but is outside the Default Pre- and Post-windows, the account is not eligible.

If the account has been billed (there is a Bill History record for the account with the same Read Date and a non-NULL Bill Time), then the account is not eligible. If there is no Bill History record with same Read Date or a Bill History Record with the same date **and** a NULL Bill Time, and the account has a scheduled read date near now, the account is eligible for billing.

> **Note:** This allows for multiple bills per month, as long as each has its own read date and the dates are further apart than the pre- and post-windows.

When an account is eligible, the system then checks its Availability.

Account Availability

An account is "available" for billing if all data needed to compute the bill is ready. There are three steps to determine if an account is available:

1. Verify that there is no Bill History record with the same Read Date and a non-NULL Bill Time.
2. Determine what data is needed.
3. Determine if it is ready.

Only if all needed data is ready is the account available for billing.

The actual process is that the program determines each data piece needed and if it is ready, data piece by data piece. The program stops immediately if any data is missing, so that it can try to bill the next account as soon as possible. You can run the Billing Exceptions Report (see **Chapter 8**: **Review Billing Exceptions** in the *Oracle Utilities Billing Component User's Guide* or **Billing Exceptions** on page 10-5 in the *Oracle Utilities Billing Component Web User's Guide*) to view all the reasons why an account might fail.

There are three types of data checked: bill determinants, interval data, and special events.

Bill Determinant Data: Bill determinant data is scalar (single-valued) data. For example, such data include simple metered kWh, and on and off peak kW from a time-of-use meter. If all the determinants needed to bill an account are computed from interval data, then the account will not require determinant data. However, some accounts require determinant data for billing.

An entry in the Bill Determinants Use (ACCTBILLDETUSE) Table indicates that the account requires the specific bill determinant for billing. If there is such an entry, the Bill History Table is read for a record with the account's read date in the Read Date column. If found, the record values are checked for the determinant. If the record or value are not present the account is rejected.

Interval Data: The Channel History (CHANNELHISTORY) Table lists the channels that are used to bill an account. If an account is eligible for billing, a list of its recorders and channels is retrieved, along with the Pre- and Post-window for each recorder. For each billed channel, the Interval Database is interrogated to make sure it has a cut with a stop date and time within the corresponding recorder's window. All such cuts must be present for the interval data to be ready.

Special Events: Special events are such things as standby, maintenance and backup schedules, opportunity demand schedules and values, interruptible times, Saturday on/off peaks, system peak time, and so on. The billing requirement is that each of these that might be used when processing a bill must be manually checked before billing can begin, even though there may not be a value for some in any given month. Part of the Review Billing Checklist (see **Chapter 4**: **Review Billing Checklist** in the *Oracle Utilities Billing Component User's Guide* or **Billing Checklist** on page 10-2 in the *Oracle Utilities Billing Component Web User's Guide*) shows the special events that must be checked for an account. The Browser or Account-Centric View can then be used to indicate which are ready. Only when all special events are ready can billing proceed.

There are three pieces of information associated with each account and special event:

1. Does the account need the event for billing?
2. Is the event ready for billing?
3. What are the actual dates and values for the event?

In the Data Repository, special events are described in the Override (OVERRIDE) Table, and tracked in the Override History (ACCTOVERRIDEHIST), Override Use (ACCTOVERRIDEUSE), and Override Ready (ACCTOVERRIDEREADY) tables. An entry in the Override Use Table indicates that an account requires a special event for billing. It is added when the account starts using it, and has a NULL Stop Time until it is no longer used for billing.

Entries in the Override History Table represent occurrences of events. These occurrences must be added to the database before billing the month they affect.

Entries in the Override Ready Table indicate that the Override is ready for billing. A separate entry is needed each month because there may not be any actual occurrences of the event.

This data can be maintained using the Account-Centric View or the Browser.

Billing Account Note/Work Queue Item Generation

This section describes how Account Notes and/or Work Queue Items are created when there is an error in the bill calculation process for Automatic or Approval Required Billing. Several types of Account Notes/Work Queue Items may be generated, and this describes all of them. In addition, there are two ways of creating the Account Notes/Work Queue Items: v2.10 and before (pre-2.10) and post-v2.10. These will also be described. You can select the style you want in the CIS Billing Options dialog (see **The Default Billing Options dialog box lets you specify default global options that apply to all accounts in the system. These options apply to all billing modules and to all Oracle Utilities Billing Component users (unlike the user-specific Tools-›Options preferences described in Chapter Two of the Oracle Utilities Billing Component User's Guide). However, you can override any of these settings for specific rate schedules, rate codes, or accounts in the Oracle Utilities Data Repository, or for individual jobs. Available Billing options include:** on page 7-2).

> **Note**: When using work queues, exceptions are written to the Work Queue Open Item table instead of the Account Notes table. To enable the Work Queues application, include the **USE_WORK_Q_FOR_ACCOUNT_NOTES = 1** parameter in the LODESTAR.CFG file.

The primary difference between pre-v2.10 and post-v2.10 is that for v2.10 and earlier, Account Notes/Work Queue Items are generated on the day after the scheduled read date, while post-v2.10 they are generated the day after the end of the postwindow from the scheduled read date. The date is called the Error Date below. In addition, when an Account Note/Work Queue Item is created in pre-v2.10 versions, the NOTETIME is the current time; in post-v2.10, it is the input Bill Date.

When the Account Note/Work Queue Open Item table is checked for Account Notes/Work Queue Items that stop billing, in pre-v2.10, the date range is from the input Bill Date minus the global pre window through the day after the input Bill Date. For post-v2.10 versions, the start date is the same, but the through date is max of the Accounts scheduled read date plus its postwindow plus a day and the input Bill Date plus a day.

> **Note:** If the input Bill Date is after the day after the post window the account will not even attempt to bill and there will be no Account Notes generated for the account. Every day should be specified as the Bill Date during at least one run.

When an Account Note/Work Queue Item was created in pre-v2.10 versions of Oracle Utilities Billing Component, the NOTETIME was the current time. In post-v2.10 versions, if the ACCOUNTNOTE/Work Queue Open Item table does not contain a READDATE column the NOTETIME is always the scheduled read date. If it does contain the READDATE column the NOTETIME is the current time and the READDATE is the scheduled read date. This change is intended to correct support for checking for Account Notes/Work Queue Items that stop billing.

> **Note:** This change does not depend on this CIS Billing Options setting.

Note that if the input Bill Date is after the day after the post window, the account will not even attempt to bill and there will be no Account Notes/Work Queue Items generated for the account. Every day should be specified as the Bill Date during at least one run.

Note that these rules apply only to non-summary customers, since their account bills may be computed long after their post window.

There are several places in the eligible/available process where an error can occur. The following tables lists the error and the result in pre-v2.10 and post-v2.10.

Error	Account Note/Work Queue Open Item Type: Pre-v2.10	Account Note/Work Queue Open Item Type: Post -v2.10
Error Reading Account Data	SYSTEM REJECTED	SYSTEM REJECTED
Previous Account Note stops billing (if Check enabled)	EXCEPTION	
Error Reading Last Bill History Record	SYSTEM REJECTED	SYSTEM REJECTED
Missing Interval Data before Error Date		
Missing Interval Data on Error Date	EXCEPTION	EXCEPTION
Missing Determinants, Overrides or Meter Values before Error Date		
Missing Determinants, Overrides or Meter Values on Error Date	EXCEPTION	EXCEPTION
Error Reading Rate Schedule	SYSTEM REJECTED	SYSTEM REJECTED
The following only apply if Account Note Check is enabled.		
Error Running Rate Schedule before Error Date	ANALYSIS	ANALYSIS
Error Running Rate Schedule on Error Date	ANALYSIS	SYSTEM REJECTED
ABORT Running Rate Schedule before Error Date	ABORT	ABORT
ABORT Running Rate Schedule on Error Date	ABORT	SYSTEM REJECTED
WARNING Running Rate Schedule before Error Date	WARNING	WARNING
WARNING Running Rate Schedule on Error Date	WARNING	SYSTEM REJECTED

Computing the Bill Start and Bill Stop

This section explains the rules for computing the Bill Start and Bill Stop date and time.

Options Overview

The following Bill Start and Stop computation rules depend on the settings in the **CIS Billing Options**, in particular the Multiple Bill History Records selection and the Interval Data settings on the Check Options tab, as well as an account's data. Bill History record dates are used if:

1. Multiple Bill History Records is set to anything but No
2. The 'Do not use Interval Data' option on the Check Options tab is checked
3. The 'Do not use Interval Data to Compute Start/Stop' options on the Check Options tab is checked
4. The account does not have any interval data used for billing. That is, it has no related Channel History records with the 'Required for Billing' column set to 'Y'.

Otherwise the interval data cut start and stop date/times are used to compute the Bill Start and Bill Stop, as described below.

Account and Channel History Start/Stop Rules with Interval Data

If Account Start is within the bill period (Bill Start <=Account Start <=Bill Stop) then use the Account Start for the Bill Start.

If Account Stop is within the bill period (Bill Start <=Account Stop <=Bill Stop) then use the Account Stop for the Bill Stop and process account like a Final Billed account (use Final Bill rules in **Bill Start and Bill Stop Computation** on page 4-15).

If Account Start and Channel History Start are within the Bill Period, use the latest one.

If Account Stop and Channel History Stop are within the Bill Period, use the earliest one.

Account Start and Stop Rules without Interval Data: Using Only Bill History Records

If Account Start is within the bill period (Bill Start <=Account Start ==Bill Stop) then issue a Warning message, but use the Bill Start.

> **Note:** A Warning message prevents Automatic Billing from billing the account, but allows Approval Required and Current Final bill to continue.

If Account Stop is within the bill period (Bill Start ==Account Stop <=Bill Stop) then issue a Warning message, but use the Bill Stop and process account like a Final Billed account (use Final Bill rules tables described in **Bill Start and Bill Stop Computation** on page 4-15).

User-Specified Stop Rules

If specified, use the User-Specified Stop for Bill Stop even if the Account Stop is within the bill period (Bill Start <=Account Stop <=Bill Stop).

If User-Specified Stop is specified, then values based on the INTDLOADLISTENERGY function (see the *Oracle Utilities Rules Language User's Guide*) are billed to the User-Specified Stop and not the cut stop.

If User Specified Stop is set in a Bill History record and interval data is used for billing, the interval data must be present up through the Bill History Stop Time. If it is not, Automatic Billing and Approval Required will not bill; you would need to use the Current/Final Bill module (see **Chapter 7**: **Current/Final Bill** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 8**:

Current/Final Bill in the *Oracle Utilities Billing Component Web User's Guide*) to bill an account with missing data.

Miscellaneous Rules

If account uses both Interval and Scalar data, use the rules for Interval data to compute the Stop Time. If the account was previously billed, use the last billed Stop Time (plus one second) as the new start. If not, use the rules for Interval data to compute the Start Time.

If a channel has multiple cuts for the bill period, then the cuts are merged into one cut for the bill period.

If an Account is switched from one option to another (such as on or off Full Day Billing), the switch only applies to the Bill Stop in the first bill period after the switch.

If an Account has multiple rates in effect and one rate is set for Full Day Billing, then all rates are treated as Full Day Billing.

If the start or stop date and time is not on a full interval boundary, round it back to the full interval boundary.

If there are multiple IPHs and not on a boundary, round back to the largest interval's boundary.

Bill Start and Bill Stop Computation

The following tables describe how the Bill Start and Bill Stop date/times are computed. The most significant factor is whether interval data is used to bill the account, but other factors include whether the account requires full day bills, and whether the account belongs to a customer that requires summary bills.

Automatic Billing

	Interval Data	Scalar Data
Regular	Use cut's earliest start (but not earlier than last months Bill History stop) Use cuts earliest stop User-Specified Stop takes precedence over cut stop	Use Bill History start Use Bill History stop User-Specified Stop takes precedence over Bill History stop
Full Day Bill	Use cut's earliest start, adjusted to previous midnight (but not earlier than last months Bill History stop) Use cut's earliest stop, adjusted to previous midnight User-Specified Stop takes precedence over cut stop	Use Bill History start (adjusted to previous midnight) Use Bill History stop (adjusted to previous midnight) User-Specified Stop takes precedence over Bill History stop
Summary Bill	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account's read date - then follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected, then only bill the specified account following the above rules.	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account's read date - then follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected then only bill the specified account following the above rules.

Approval Required

	Interval Data	Scalar Data
Regular	Use cut’s earliest start (but not earlier than last months Bill History stop) Use cut’s earliest stop User-Specified Stop takes precedence over cut stop	Use Bill History start Use Bill History stop User-Specified Stop takes precedence over Bill History stop
Full Day Bill	Use cut’s earliest start, adjusted to previous midnight (but not earlier than last months Bill History stop) Use cut’s earliest stop, adjusted to previous midnight User-Specified Stop takes precedence over cut stop	Use Bill History start (adjusted to previous midnight) Use Bill History stop (adjusted to previous midnight) User-Specified Stop takes precedence over Bill History stop
Summary Bill	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account’s read date. Follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected, then only bill the specified account following the above rules.	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account's read date. Follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected, then only bill the specified account following the above rules.

Current Bill

	Interval Data	Scalar Data
Regular	Use cut's earliest start (but not earlier than last months Bill History stop) Use cut's earliest stop User-Specified Stop takes precedence over cut stop	Use Bill History start Use Bill History stop User-Specified Stop takes precedence over Bill History stop
Full Day Bill	Use cut's earliest start, adjusted to previous midnight (but not earlier than last months Bill History stop) Use cut's earliest stop, adjusted to previous midnight User-Specified Stop takes precedence over cut stop	Use Bill History start, adjusted to previous midnight Use Bill History stop, adjusted to previous midnight User-Specified Stop takes precedence over Bill History stop
Summary Bill	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account's read date. Follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected, then only bill the specified account following the above rules.	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are held until the latest account's read date. Follow the above rules for each account. If the **Include Related Summary Accounts** option is not selected, then only bill the specified account following the above rules.

Final Bill

	Interval Data	**Scalar Data**
Regular	Use cut's earliest start (but not earlier than last months Bill History stop) Use cut's latest stop (use all available interval data) User-Specified Stop takes precedence over cut stop	Use Bill History start Use Bill History stop User-Specified Stop takes precedence over Bill History Stop
Full Day Bill	Use cut's earliest start - adjusted to previous midnight (but not earlier than last months Bill History stop) Use cut's latest stop - ignore Full Day Bill (use all available interval data) User-Specified Stop takes precedence over cut stop (do not adjust to midnight)	Use Bill History start (not adjusted to midnight) Use Bill History stop (not adjusted to midnight) User-Specified Stop takes precedence over Bill History stop (do not adjust to midnight)
Summary Bill	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are final billed following the above Final Bill rules. If the **Include Related Summary Accounts** option is not selected, then only final bill the specified account following the above Final Bill rules.	If the **Include Related Summary Accounts** option is selected, then all accounts for the bill month are final billed following the above Final Bill Rules. If the **Include Related Summary Accounts** options is not selected, then only final bill the specified account following the above Final Bill rules.

Bill Correction

	Interval Data	Scalar Data
Regular	Use cut's earliest start (but not earlier than last months Bill History stop) Use cut's earliest stop User-Specified Stop takes precedence over cut stop (use Bill History stop if the User-Specified Flag = Y in Bill History Table)	Use Bill History start Use Bill History stop User-Specified Stop takes precedence over Bill History stop (use Bill History stop if the User-Specified Flag = Y in the Bill History Table)
Full Day Bill	Use cut's earliest start - adjusted to previous midnight (but not earlier than last months Bill History stop) Use cut's earliest stop, adjusted to previous midnight User-Specified Stop takes precedence over cut stop (use Bill History stop if the User-Specified Flag = Y in the Bill History Table)	Use Bill History start (<u>not</u> adjusted to midnight) Use Bill History stop (<u>not</u> adjusted to midnight) User-Specified Stop takes precedence over Bill History stop (do <u>not</u> adjust to midnight) (use Bill History stop if the User-Specified Flag = Y in the Bill History Table)
Summary Bill	If the **Include Related Summary Accounts** option is selected, then all accounts are corrected following the above Bill Correction rules for each account. If the **Include Related Summary Accounts** option is not selected, then only correct the specified account following the above Bill Correction rules.	If the **Include Related Summary Accounts** option is selected, then all accounts are corrected following the above Bill Correction rules for each account. If the **Include Related Summary Accounts** option is not selected, then only corrected the specified account following the above bill Correction rules. If the **Use Bill History Dates and Times** option is selected, Oracle Utilities Billing Component uses the Start and Stop dates in the Bill History record.

Summary Bill Calculation

This section describes summary rate schedules. The next section (**Non-Summary Bill Calculation Process**) describes the steps that occur to execute a rate schedule in Oracle Utilities Billing Component. This is important because a rate schedule must be designed correctly to get the proper results, and the execution process is where the results are determined. The purpose of this section is to help you picture the processing so you can more easily understand how a rate form will work. Following sections describes the overall run process, with the enhancements. Some steps are elaborated in following sections.

The processes outlined here support single bill calculations as well as many types of summary bills. Oracle Utilities Billing Component supports the following multiples, which are combined into summary bills. The order listed here is important, since the processing is done in this order.

- Per Account

  1.Multiple rate schedules per bill history record - inner loop

  2.Multiple bill history records per read date - outer loop.

- Per Customer if Summary Customer

  1.Each Account belonging to the Customer - outermost loop.

The following sections assume one "master" account per summary customer. The master account will be the last one associated with a customer summary rate schedule on or before the current date (see below).

Non-Summary Bill Calculation Process

Note: The following assumes the accounts are all ready to bill.

When one of the three Bill Calculation modes, Automatic, Approval Required, and Current/Final Bill, start with an account, they first determine if it is part of a Summary Customer. If so, all related accounts are retrieved and processed one after the other. Then, for each summary account, all its bill periods in the bill month are determined (based on read dates in the Billing Cycle Date Table).

If an account is not part of a summary customer, its next unbilled read date is determined.

At this point Oracle Utilities Billing Component is ready to bill one account for one bill period. It determines the overall bill period start (from interval data or the first bill history start if multiple bill history records) and stop (from interval data or the last bill history stop if multiple bill history records). It then finds all rate schedules that were in effect at any time in this period (via the Rate Code History Table).

Next, for each of the bill history records, Oracle Utilities Billing Component runs all rate schedules where the bill history start/stop dates overlap the rate schedule in effect dates (if multiple rate schedules allowed, otherwise it uses the most recent one - by Start Time). Determinant values are loaded from the bill history record (if available).

After each rate schedule is run its results are "accumulated". There is an "accumulator" for each possible summary level. The current value of determinants and revenue identifiers is added to their corresponding accumulators (determinants are added once - for the first rate schedule - while the revenue from each rate schedule is accumulated). The accumulated bill start and stop times are computed (earliest Start/latest Stop per level).

The next to last step for a rate schedule is to make the current values available for reporting. The final step is to reset the values in preparation for the next bill calculation (see **Shared Symbol Table** below).

Shared Symbol Table

When a rate schedule is prepared for use in a bill calculation, its identifiers are put into a symbol table. The symbol table holds the identifier name, label, value, historical values, and other information related to the identifier. All rate schedules run during a bill calculation share a common symbol table. This is necessary so that the "accumulate" step above will accumulate across multiple rate forms. However, this means that the identifier labels are also shared. The value of these will be that of the last rate form to assign them. In other words, the analysis reloads and re-uses the same symbol table.

In order for the accumulation of individual results into summaries to work, identical values must be named the same. For example, $KWH_CHARGE should be used consistently as the energy charge in all rate forms. Similarly the total revenue identifier must be the same in all rate forms.

Revenue symbols are displayed in reports in the order they are entered into the symbol table. They are entered in the order they first appear in the rate forms, from the first to the last. This means that a revenue symbol in a later rate form that is not in the first may appear after other revenue symbols that it is before in this later rate form, because they were used in an earlier rate form. To avoid this you should create a rider that uses the REVENUE statement to define all revenue symbols that appear in any rate form, and INCLUDE it at the beginning of all rate schedules.

Oracle Utilities Billing Component 2.10 Bill Calculation

The following pseudocode outlines the processing steps involved in bill calculations in version 2.10.xxx of Oracle Utilities Billing Component.

```
FOR EACH Account in input list
        Get all its related summary accounts, or just itself if none
        FOR EACH Related Account
              Get all its bill periods in this bill month, or just current if not summary customer
              FOR EACH Bill History record
                    Determine its bill start and stop data/times
                    Get a list of all rate schedules in effect at any time in the period
                    FOR EACH Rate Schedule
                          Process the Rate Schedule
                          Accumulate revenue and determinant values
                          Display a page with the results
                          Reset values
                    END FOR
                    If there was more than one rate schedule processed
                          Display a page of the summary results
                    END IF
              END FOR
              If there was more than one bill history record processed
                    Display a page of the summary results
              END IF
        END FOR
        If there was more than one related account processed
              Display a page of the summary results
        END IF
END FOR
Display a page of the grand total results
```

Oracle Utilities Billing Component 2.11+ Bill Calculation

The following pseudocode outlines the processing steps involved in bill calculations in version 2.11.xxx and later of Oracle Utilities Billing Component.

```
FOR EACH Account in input list
    Get all its related summary accounts, or just itself if none
    FOR EACH Related Account
        Get all its bill periods in this bill month, or just current if not summary customer
        FOR EACH Bill History record
            Determine its bill start and stop data/times
            Get a list of all rate schedules in effect at any time in the period
            FOR EACH Rate Schedule
                Process the Rate Schedule
                Accumulate revenue and determinant values
                Display a page with the results
                Reset values
            END FOR
            If account summary rate schedules exist at the rate schedule level
                Run them and display each results
            ELSE
                If there was more than one rate schedule processed
                    Display a page of the summary results
                END IF
            END IF
        END FOR
        If account summary rate schedules exist at the bill history level
            Run them and display each results
        ELSE
            If there was more than one bill history record processed
                Display a page of the summary results
            END IF
        END IF
    END FOR
    If customer summary rate schedules exist
        Run the customer summary rate schedule
    ELSE
        If there was more than one related account processed
            Display a page of the summary results
        END IF
    END IF
END FOR
Display a page of the grand total results
```

Summary Rate Schedules

After each loop a summary rate schedule may be run. Oracle Utilities Billing Component needs to know is a rate schedule associated with an account is a summary rate schedule, and if so its level. This is done via the Rate Code. Three summary rate codes are supported:

- "SUMMARY_RS" - Summarizing multiple rate schedules for one bill period for one account
- "SUMMARY_BH" - Summarizing multiple bill history records for one bill period for one account
- "SUMMARY_CUST" - Summarizing multiple accounts for one customer. Only one in effect at a time for all accounts of a summary customer. The most recent (via the Start Time in the Rate Code History record) account with this rate code is the "master account" and will be billed last when billing a summary customer.

Summary Bill Calculation Process

If enabled, a summary rate schedule is run after all other bill calculations are performed at a level. Note that a summary is run regardless of the number of bill calculations at this level or below.

All summary rate schedules are run in the context of an account, using the shared symbol table. The account's bill history record is loaded first. Then accumulator values at the appropriate summary level are copied to become the current values. The current bill dates are set from their accumulated values.

1. The BILL_START identifier will be set to the earliest BILL_START of any summarized bill calculation. The BILL_STOP identifier will be set to the latest BILL_STOP of any summarized bill calculation. BILL_PERIOD and READ_DATE will also be set to their latest values.
2. All revenue and determinant identifiers referenced in a summary rate schedule will be set to their appropriate accumulated values before running the summary rate schedule.

When the rate schedule is run it can perform calculations, save CIS records, and save to the database. If it modifies a revenue amount or determinant, the change will be propagated to the accumulated values at higher levels.

If a summary rate schedule is run and reported, there is no separate standard summary report (the standard summary report is run automatically only if there is no summary rate schedule).

Summary Customer Account Iteration

Within a customer summary rate schedule, there may be a need to perform some operation on each of the summary accounts. Because the rate schedule is run in the context of an account, CUSTOMER.CUSTOMERID is the ID of the corresponding customer record. A simple list will return all accounts with the same customer record.

Summary Account Charges

When an account posts a charge, it must be posted to the “master” account. The master account is defined as the one with a SUMMARY_CUST rate code. The master account UID is saved internally while a rate schedule runs. Charges will automatically post to it. In addition, if needed, the master account's current read date will be saved and used.

The account ID for the master account is provided in the MASTER_ACCOUNTID identifier. If there is no master account, this identifier will not have a value.

Summary Options

Each summary rate schedule is associated with an account, even the customer summary rate schedule. Since all other rate schedules must run before the customer summary rate schedule, the account it is associated with must be the last account to run. If Customer level summary rate schedules are enabled Oracle Utilities Billing Component, will automatically order the accounts so the one with the customer summary rate schedule will run last (it is an error to have two such in effect at the same time for different account for the same customer). Since it takes time to do this check, the customer summary rate schedule feature may be disabled. The option is set on the Summary Options tab of the Default Billing Options dialog (see **Default Billing Options** on page 7-2).

Enabling summary options slows the billing process, even if the option is not used, because Oracle Utilities Billing Component must still check for appropriate data.

Saving the Bill History Record

During the bill calculation, determinants and the Bill Stop date are calculated. These are saved to the Oracle Utilities Data Repository after the bill is approved. During a re-bill these values may be modified, and the results also saved. This section describes the “rules” for saving Bill History records.

Saving or writing a Bill History record means writing a record to the Bill History (BILLHISTORY) Table, and writing a record or records to the related Bill History Value (BILLHISTORYVALUE) Table, as determined by the value of the BILLHISTORY Column (BILLHISTCOLNAME) in the Bill Determinants (BILLDETERMINANT) Table. It may also mean adding records to the Bill History Edit (BILLHISTORYEDIT) and Bill History Edit Value (BILLHISTORYEDITVAL) tables if existing data is changed.

Rules

There are two possibilities: either there is no overlap (based on STARTTIME and STOPTIMEs) with another record for the same account, or there is some overlap. If there is no overlap, the system simply writes the record. The following rules apply when an overlap occurs.

> **Note**: These rules do not apply to small overlaps, where the STOPTIME for one record is the same as the STARTTIME for the next. These rules apply to larger overlaps.

1. If the only overlap is at the beginning of the new record, and it is only for the start day, the system ignores it and writes the record. It also writes the record if the overlap is at the end of the stop day.

   If the new record has exactly the same STARTTIME as an existing record, then the new one simply replaces the old one:

2. If the STARTTIME matches an existing record's start time, the system reads the old record, saves changed fields to the Bill History Edit tables, and updates the Bill History and Bill History Value records, all in one transaction. Note that the update may remove or modify Bill History Value records.

   If there is a big overlap (more than a day) at the beginning of the new record, there is an error. This means that the previous month's record is wrong (its STOPTIME is wrong, and it’s likely the determinants are too). We will allow for the two possibilities in rule 3:

3. If there is a large beginning overlap and the user specifies the BH_CORRECT_PREVIOUS_STOPTIME flag, then the system updates the previous month's STOPTIME, making it the new STARTIME (as in rule 2 above), and writes the new record. If the user does not specify this flag, the system returns an error and an appropriate error message.

   There may also be an overlap at the end of the record. If it is one day, it is ignored and the new record is written. If it is greater than a day, it must be changed manually. This can be addressed with a re-bill after several months. In this case, the user should then re-bill the next month as well.

4. If there is a large end overlap and the user specifies the BH_CORRECT_NCXT_STARTIME flag, then the system updates the next month's STARTTIME (making it the new STOPTIME as in rule 2 above), then writes the new record. If the user does not specify this flag, the system returns an error and an appropriate error message.

   An old record's dates may be totally inside the new one's, or totally outside the new one's. This leads to:

5. If an old record is totally contained in a new one or completely contains a new one, then the system deletes the old one (writes to edit trail first) and proceeds as above.

Saving Data from a Rate Schedule

This section explains how data is saved from a rate schedule. This includes:

- An overview of the types of data that can be saved from a rate schedule
- An overview of how bill page/rate schedule runs are related
- A description of the requirements for saving data from a rate schedule
- A description of how transactions are processed, and the specific types of data that can be saved via a "Postponed Save" (after user-approval)
- A description of a solution for saving all types of data from a rate schedule.

Types of Data

There are several distinct types of data that can be saved from a rate schedule. These include:

- Relational Database Records (can be added, updated, or deleted)
- Interval Data Cuts (can be saved or deleted)
- Financial Charges
- Billing Determinants
- Relational Database Column Updates (via the LISTUPDATE function)
- CIS Records.

In addition, Account Notes/Work Queue Items may need to be written during a bill calculation.

Related Bill Pages

In all billing modes (Automatic, Approval Required, Current/Final, Bill Correction and Trial Bill/Report), related bill pages and/or rate schedule runs are grouped. One bill page represents one bill history record processed by one rate schedule. Bill pages are related and grouped when:

a) They are all for accounts that have the same summary customer

b) They are for the same account, with the same scheduled read date. These are done in a nested loop. The outer loop is by bill history record; the inner is by rate schedule.

c) They are for all bill periods to be corrected (for Bill Correction only).

In previous versions of Oracle Utilities Billing Component, a group could only be approved all together, and then only if there were no errors in any bill page. The current version of Oracle Utilities Billing Component includes an option to allow for approval of all pages that did not have an error.

Requirements

There are several requirements that affect how data can be saved:

1. All changes a rate schedule makes to either database (Data Repository or interval database) must be available to later, related rate schedule runs.
2. The user must have to capability of approving or rejecting ALL saves, except in Automatic bill mode.
3. For Automatic and Approval Required billing, account notes/work queue items may be written during the billing process.
4. Interval data saves and deletes should be made only after user approval (see 2, above). In previous versions, interval data saves and deletes occurred immediately.

Transactions

All changes to the relational database take place in the context of a transaction. All changes within a transaction are committed together, or all are rolled back (removed) together. In all billing modes, all related rate schedules are run in a single transaction that is started before the first rate schedule in the group is run. After the last rate schedule in the group is run there are several possibilities:

1. If there were any errors, roll back all saves.
2. If the user will approve the group later, roll back the changes and then redo them when approved (it is very bad database practice to have a transaction open waiting for user action).
3. When using RUNRS or Automatic Billing, commit the saves and then write out CIS records.

These possibilities do not allow for saving of interval data stored in the Oracle Utilities Data Repository.

Postponed Saves

To support user approval, some of the data to be saved is stored in the report data structure (the in-memory copy of the report). The data stored there includes:

- Relational Database records
- Billing Determinants
- CIS records.

No other type of saved data is stored for user approval. This means that all the other data must be committed after a run or group of runs. In prior releases, interval data was stored in the Btrieve database, and was therefore exempt from relational database commits. LISTUPDATE was only used by RUNRS, and was committed after each run.

> **Note:** When the SAVE TO TABLE statement runs, records are usually written - temporarily - to the relational database to verify the correctness of the record. See the NO_TEMP_SAVE configuration parameter for the exception.

Note that interval data, XML data, and financial charge updates stored in the Oracle Utilities Data Repository are not exempt from the relational database commits. To save them (do a commit), all other database changes will also be committed. This contradicts the requirement for user approval of saves.

Two Phase Commit

The solution is as follows:

1. All billing modes have additional approval choices:
    - Automatically save/approve group of related pages is all is OK, and
    - Automatically save/approve each page if it is okay.

   Using either of these options will commit all saves and approve the bill if there are no errors.
2. If the rate schedule(s) are expected to save/delete interval data in the relational database, use the LISTUPDATE function, or possibly modify financial data, and the user wants to approve the data, the user must:
    - Run the bill with Enable user approve/reject of saves, and
    - If all is OK, approve the bill. The bill will be rerun (in the same billing mode) with Automatic Approval.

This second run will generate the same results, and save all the data.

Multiple Bill History Records per Bill Period

This section describes the support for multiple Bill History records per bill period. There are multiple Bill History records in a bill period if two or more BIll History records have the same value in the Read Date column (READDATE). Support for this includes sub-period billing, partial reads, out of order bills, and Oracle Utilities Rate Management support for all of the above.

Sub-Period Billing: In some case many billing factors may change during a bill period. These include prices, temperature constraints, regulatory changes, etc. Oracle Utilities Billing Component needs to report the bill amount for every time period where all factors stay the same. Each of these "small" bill periods starts when one or more factors change, and ends when another set of factors changes. The bill amount for the entire bill period is the sum of the bill amounts for the small bill periods.

The user must create determinants for each small bill period. While processing a small bill period a rate schedule may require access to the entire bill period's interval data.

The applications save each set of determinants as one Bill History record in the database. All Bill History records for a bill period will be related by sharing a common READDATE. Each Bill History record can then be processed using the usual bill calculation process.

Partial Reads: Users may receive partial scalar data from Meter Reader Service Providers (MRSPs). Each is stored as a Bill History record, with a common READDATE. These scalar values must be aggregated to the bill month level to form the determinants for the month.

Out-Of-Order Reads: Users may get out-of-order scalar reads. Thus they may skip July, bill August, and then get July's and September's data at the same time. These last two are stored as two Bill History records with the same READDATE.

Oracle Utilities Rate Management/Data Manager: The Customer Revenue and Customer Impact analyses in Oracle Utilities Rate Management, and the Bill History Report in the Data Manager, all deal with monthly data. If there are multiple Bill History records in a bill month, they must be aggregated for these analyses and reports to work.

Requirements

Oracle Utilities Billing Component:

1. Supports multiple sets of determinants (bill history records) per bill period (a bill period is designated by a read date).
2. Aggregate determinants to a read date or bill month.
3. Computes a "bill" for each set of determinants.
4. Shows the user the results of all the individual computations.
5. Exports to CIS results of all individual computations.
6. Supports for multiple read dates per bill month, multiple rate schedules per account, and multiple accounts per customer.

Multiple Bill History records per bill period applies only to scalar data. If there is also interval data, then the start and stop dates in the Bill History records will override the cut start and stop dates when determining a bill periods' start and stop dates.

Cancel/Rebill is not be supported when using multiple bill history records per bill period.

Aggregation Requirements

In order to create a summary report, data in individual reports are aggregated to different levels, or combined to create the summary values. Oracle Utilities Billing Component supports the following aggregations (in ascending order of bigger aggregations):

1. Multiple bill history records to the bill period level (for an account),

2. Multiple bill periods to the bill month level (for an account),
3. Multiple rate schedules to the account level,
4. Multiple accounts to the customer level (per bill month), recursively.

All bill history records in the same bill period will have the same READATE. This groups the records together, with each record representing one sub-period of the bill period. Different bill periods have different read dates.

In addition, to support Oracle Utilities Rate Management analyses, the bill calculation engine also supports:

5. Multiple bill months per analysis period.

Item #3 above (multiple rate schedules to the Account level) implies that each rate schedule is applied to each bill history record/bill period in #s 1 and 2 that is in its effective date range. In other words, each sub-period will be billed by each rate schedule that is in effect during the sub-period.

Aggregation Values

There are three types of Rules Language data that are aggregated: bill dates (BILL_START and BILL_STOP), revenue values and determinants. For dates, the earliest BILL_START date and latest BILL_STOP date in an aggregation are used. Revenue values are always added. However, the rules for determinant aggregation are more complex. This section lists the rules for aggregating determinants for each multiple above:

Determinants: To get a determinant's value for a bill period or bill month, from multiple bill history records or bill periods, the determinant's values are combined based on the Aggregate flag on the Bill Determinant table (see below). Averages are weighted based on the number of seconds covered by a bill history record, compared to the total number of seconds in the bill period (or number of seconds in a bill period compared to the number of seconds in a bill month).

Rate Schedules: With multiple rate schedules in a bill period the following assumption is necessary:

- Determinants with the **same** name in different rate schedules represent the **same** value.

  So, when aggregating multiple rate schedules there should be only two cases -

  a. The determinant was used in a preceding rate schedule and its current value matches the accumulated value,

  b. The determinant was not previously used, so copy the current value as the accumulated value.

If the current value does not match the accumulated value, Oracle Utilities Billing Component generates a warning message.

Note: As of version 2.02.159, if the determinant does not match the accumulated value, it is added.

To aggregate determinants from multiple accounts or multiple bill months, the values are added in based on the Totalize flag in the Unit-of-Measure table.

Rules Language identifiers are considered determinants:

1. It matches the IDENTIFIER value of a record in the BILLDETERMINANT table,
2. It is the identifier in a DETERMINANT statement, or
3. It is the determinant in an ALL or BLOCK statement.

Derived Requirements

1. Older versions of Oracle Utilities Billing Component supported at most 36 values for historical determinants (three years worth). Here there may be as many as one set of determinants a day. Hence, to capture three years of historical data Oracle Utilities Billing Component supports at least 366 X 3 values for historical input determinants.
2. To use these historical values they usually must be converted to values on a per month basis. The Oracle Utilities Rules Language in Oracle Utilities Billing Component supports the aggregation of determinants to monthly values. It also supports all historical and season functions using the aggregated determinants.
3. Oracle Utilities Billing Component "knows" the Unit-of-Measure and Aggregate flag for every determinant and uses it for aggregation rules. Determinants from the database have these designated in the Bill Determinant table.

Functional Specifications

This section lists specific components of Oracle Utilities Billing Component that support the above requirements.

Bill Determinant Table

The Aggregate column in the table describes how to combine multiple determinant values. If it is NULL then the Aggregate flag on the determinant's corresponding record in the Unit-of-Measure (UOM) table is used. This is needed because the UOM Aggregate flag for kW is A (average), while it should be M (maximum) when combining multiple reads in a bill month.

Shared Symbol Table

When a rate schedule is prepared for use in a bill calculation, its identifiers are put into a symbol table. The symbol table holds the identifier name, label, value, historical values, and other information related to the identifier. All rate schedules run during a bill calculation share a common symbol table. This is necessary so that the "accumulate" step will accumulate across multiple rate forms.

In other words, the analysis loads input values into and re-uses the same symbol table for each sub-period and rate schedule.

In order for the accumulation of individual results into summaries to work, identical values must be named the same. For example, $KWH_CHARGE could be used consistently as the energy charge in all rate forms. Similarly the total revenue identifier must be the same in all rate forms.

Revenue symbols are displayed in reports in the order they are entered into the symbol table. They are entered into the symbol table in the order they first appear in the rate forms, from the first to the last. This means that a revenue symbol in a later rate form that is not in the first may appear after other revenue symbols that it is before in this later rate form, because they were used in an earlier rate form. To avoid this you should create a rider that uses the REVENUE statement to define all revenue symbols that appear in any rate form, and INCLUDE it at the beginning of all rate schedules.

Similarly, there should be one rider that contains all DETERMINANT statements. It would also be included at the beginning of all rate schedules.

Rules Language Functionality

1. Every identifier is compared to those in the Bill Determinant table to determine if the identifier is a determinant. If so, its UOM and Aggregate flags are loaded from the corresponding database records.
2. Oracle Utilities Billing Component supports up to 1200 values for an input historical determinant. This allows for three years of data, with one bill history record per day (for many sub-periods). A computed determinant is restricted to 60 values.

3. Oracle Utilities Billing Component supports all historical and season functions using the aggregated determinants described below.
4. Oracle Utilities Billing Component supports saving a determinant whose name is the value of another identifier (via the @ operator).
5. Two identifiers, FULL_BILL_START and FULL_BILL_STOP will describe the entire bill period. For non-sub-period billing these will be the BILL_START and BILL_STOP values, respectively. For sub-period billing they will be the start and stop of the entire bill period. They can be used with INTDLOADDATES to retrieve a cut for the entire bill period.

Bill Calculation Process

1. Bill calculation processing includes multiple bill history records per bill period.
2. To use historical values from sub-periods, they must be converted (aggregated) to values on a per month basis. Determinants will be aggregated based on the determinant Aggregate flag (usually TOTAL or MAXIMUM). The minimum BILL_START and maximum BILL_PERIOD and BILL_STOP dates are used.
3. For sub-period billing, the bill calculation processes one sub-period at a time, from the oldest to the most recent. The “current” bill period will be the sub-period, and the previous month values will be those from the last full bill period, skipping all previous sub-periods in the current full bill period.
4. All related bills are approved together.

Multiple Bill History Records Options

Some of the steps in the Bill Calculation Process are optional. The options are set on the **Multiple Bill History Records** tab of the Default Billing Options dialog.

Enabling one of the multiple bill history record options slows the billing process, even if the option is not used, because Oracle Utilities Billing Component must still check for appropriate data. These options mean:

- **No**: If two or more Bill History records have the same READDATE the following error message will be reported in Oracle Utilities Billing Component CIS Billing modules:

  "PLM0579: Multiple Bill History records per bill period not enabled. Check data and CIS Billing Options."
- **Bill Individually - Allow Out of Order Bill Periods:** There can be two or more Bill History records with the same READDATE, each is treated as an entire bill period. There is no aggregation of determinants or dates. All records before the current one are treated as historical bill periods. All records with NULL BILLTIME are retrieved, the records are billed from the first to the last based on STARTTIME. In Oracle Utilities Billing Component, SAVE'd determinants are written back to the individual records.
- **Bill Individually - Monthly Merge Historical Values:** This is for sub-period billing. There can be two or more Bill History records with the same READDATE, each is treated as a small entire bill period. The records before the current one and with its READDATE are ignored. Records before the current READDATE have their determinants and dates aggregated to monthly values. In Oracle Utilities Billing Component, SAVE'd determinants are written back to the individual records.
- **Aggregate to Read Date and Bill As One:** This is one version of partial reads. This bills each scheduled read date, and allows for two or more scheduled read dates per bill month, and two or more Bill History records per read date. All Bill History record determinants and dates are aggregated to the READDATE (records with the same READDATE are combined). In Oracle Utilities Billing Component, SAVE'd determinants are written back to the latest record with the current READDATE. All records with the current READDATE are marked as billed (BILLTIME set).

- **Aggregate to Bill Month and Bill As One:** This is the second version of partial reads. This bills an entire bill month, yet allows for two or more scheduled read dates per bill month, and two or more Bill History records per read date. All Bill History record determinants and dates are aggregated to the BILLMONTH (records with the same BILLMONTH are combined). In Oracle Utilities Billing Component, SAVE'd determinants are written back to the latest record with the current BILLMONTH. All records with the current BILLMONTH are marked as billed (BILLTIME set).

  If the selected option is not **No**, all Oracle Utilities Rate Management analyses and the Data Manager Bill History Report set it (temporarily) to this value before performing their calculations.

Invoice Numbering

This section summarizes the requirements and support for invoice numbers in Oracle Utilities Billing Component.

Definitions

The following defines the various terms used below.

Enterprise: The entire collection of business units that are related through billing, invoicing, etc.

Billing Determinant: A single value that represents some usage measurement. It, along with a price, is used in a formula to produce a charge for the usage.

Account: A point of metered usage (or a collection of metered points, if the metered values are combined to create a single billing determinant). A bill is for an account, even if it is not sent to the account. "Account" always refers to a Oracle Utilities Billing Component account.

Bill: The displayed or printed results of running a bill calculation. A bill that is not an invoice (see next item) is an informational bill. A bill may be a summary bill that contains a summary of several other bills.

Invoice: A bill sent to a customer that legally notifies the customer of an amount due. Also called the "Actual Bill" or "Actual Invoice". A bill may also be sent to the customer for informational purposes (a Memo Bill), only an invoice requires payment. An "invoicee" is a billing entity receiving an invoice.

Invoice Numbering Requirements

- Each invoice may optionally have an invoice number.
- The invoice number may be unique for each account, or common across all invoices sent to one invoicee. This is determined by a billing entity parameter.
- If an account has multiple rate schedules, all billed at once, they will share the same invoice number.
- It is necessary to be able to look up an account based on the invoice number.
- Invoice numbers must be unique for all Operating Companies and for a long time.
- Invoice numbers may have an alphabetic prefix to help uniqueness, however that is optional.
- Invoice numbers may have an alphabetic suffix to help uniqueness, however that is optional.
- The first invoice number may be set by the user, the default first invoice number is one.
- Invoice numbers are by default automatically generated by Oracle Utilities Billing Component.
- Generated invoice numbers must be unique. Invoice numbers generated against the same database must be sequential, numbers may not be sequential if two or more databases are involved.
- An invoice number is only generated if a bill is approved; it is not generated for rejected, error or informational bills.
- An invoice number (in any string format) may specified in a Rate Schedule, if specified there it prevents automatic invoice number generation. If the specified invoice number contains a substring that is ##### (five #s) the substring will be replaced by a generated invoice number.
- The invoice number may be optionally part of any CIS record.

Invoice Numbering Database Schema

This section explains the two tables in the database, the Invoice Number table and the Bill History Invoice Number table that support invoice numbering. These tables are not part of the standard schema. To get them contact Oracle Customer Support.

Invoice Number

Records in the Invoice Number table represent different invoice types and formats. This allows for multiple invoice numbering schemes. This table includes the following columns:

- **Invoice Name**: Each record is identified by its name (INVOICENAME).
- **Prefix**: *Optional.* If populated, the string will be prefixed to each invoice number.
- **Suffix**: *Optional.* If populated, the string will be appended to each invoice number.
- **Width**: Indicates how many numeric characters to generate. A small invoice number will be prefixed with zeros to fill the width. A number that is too large will be left truncated. The default value is 8.
- **Number**: Contains the most recent invoice number (NULL means 0). It is updated when each invoice number is generated. A user can initialize it to an appropriate value before the first bill is generated.

Specifying the Default Invoice Name

The default Invoice Name is set using the INVOICENAME=X configuration file parameter, where X is the value of the INVOICENAME column. See **LODESTAR.CFG** on page 2-2 in **Chapter 2**: **Configuration Files** in the *Oracle Utilities Energy Information Platform Configuration Guide*. The default name if there is no configuration value is LODESTAR.

You can also specify an Invoice Name at the rate form level through use of the LSRSENV.INVOICENAME Rate Schedule Environment identifier. See **Rate Schedule Environment Identifiers** in **Chapter 4**: **Identifiers, Constants, and Expressions** in the *Oracle Utilities Rules Language User's Guide*.

Bill History Invoice Number

The Bill History Invoice Number table relates an invoice number to a Bill History record, and to an account. Over time one Bill History record may have several invoice numbers as the account is rebilled. The table contains three columns: UIDBILLHISTORY, BILLTIME and INVOICENUMBER - which are all part of the key. The INVOICENUMBER column is the invoice number as it appears in the CIS records.

CIS Format File Invoice Numbering Support

The keyword [INVOICENUMBER] at the top of the CIS format file (CISFORMT.TXT) indicates that approved bills require invoice numbers.

A field label INVOICE_NUMBER PIC X will be filled with an invoice number that obeys the requirements above. The width of the field may be specified - X(10) for example. This should be the same as the length of the PREFIX string plus the WIDTH from the Invoice Number table. If it is not, the invoice number is left-truncated to fit.

Within one account the invoice number will be the same whenever it is used. It will also be the same for all accounts of a summary customer.

Example

A sample part of a CIS format file supporting invoice numbers is:

```
[INVOICENUMBER]
...
[SECTION]  CUSTINVOICE
```

```
INTFC_CUST_ID;  PIC X
BILLDATE;       PIC X
INVOICE_NUMBER  PIC X
```

The following Rules Language would be placed at the top of each rate schedule:

```
/* Create the invoice number record */
INVREC.BILLDATE = CURRENT_DATE;
SAVE INVREC TO CIS SECTION "CUSTINVOICE";
...
```

If the user wants to create their own invoice number, for example containing a letter, the bill year and month, and the account id, an additional line before the SAVE could create it. For example:

```
INVREC.INVOICE_NUMBER = "A-" + YEAR(BILL_PERIOD) + "-" +
MONTH(BILL_PERIOD) + "-" + ACCOUNT.ACCOUNTID;
```

Using Automatic Billing with Oracle Utilities Meter Data Management

Automatic billing (Autobill) can be configured to work with usage data stored in the Oracle Utilities Meter Data Management tables rather than the Channel Cut Header and Meter Read tables used by default.

This section outlines how to use the Autobill feature of Oracle Utilities Billing Component with Oracle Utilities Meter Data Management, including:

- **Enabling Autobill to work with Oracle Utilities Meter Data Management**
- **Checking Account Eligibility**
- **Checking Data Availability**
- **Calculating the Bill Start and Bill Stop**

Enabling Autobill to work with Oracle Utilities Meter Data Management

By default, using data stored the Oracle Utilities Meter Data Management tables is disabled, and must be enabled on the Check Options tab of the Billing Options screen (or dialog).

How to enable autobill to work with Oracle Utilities Meter Data Management (web):

1. Select **Tools and Utilities->Options->Billing Options**.

   The **Billing Options** screen appears.
2. Select the **Check Options** tab.
3. Check the **Use MDM schema for Billing** checkbox.

How to enable autobill to work with Oracle Utilities Meter Data Management (C/S):

1. Select **CIS Billing (or Analysis)->Billing Options**.

   The **Default Billing Options** dialog box appears.
2. Select the **Check Options** tab.
3. Check the **Use MDM schema for Billing** checkbox.

Checking Account Eligibility

When using Autobill with Oracle Utilities Meter Data Management, Autobill checks for eligible accounts in the same manner as described under **Account Eligibility** on page 4-8, with one exception:

- The bill date must also fall within the Pre and Post Window of the Stop Time on the Service Point Account Record for the account.

Checking Data Availability

When using Autobill with Oracle Utilities Meter Data Management, Autobill checks for data availability based on the following guidelines:

Selecting the Meter(s) for the Account

For each account to be processed, Autobill does the following:

- Selects the Service Point(s) associated with the account from the Service Point Account table,
- Selects the meter associated with each Service Point from the MDM Meter table, and
- Determines if usage for each meter is required for billing from the Billed? flag on the Meter Configuration table, and

Checking for Available Usage

Once an account is determined to be eligible for billing, Autobill next checks for available data for the account.

When checking for available usage data, Autobill checks for available usage in either the Meter Data Channel Cut table (for interval usage), or the Meter Data Reading table (for consumption and/or time-of-use usage) based on the meter(s) selected for the account.

To be considered "available" for billing, usage data must meet the following requirements:

- The reading must have a Reading Category of "Final,"
- The reading must have a Reading Status of "Valid,"
- The reading must fall between the Start Time and Stop Time on the Account record
- The reading must fall between the Start Time and Stop Time on the Service Point Account record for the account.

Usage for a meter with any other Reading Category or Reading Status, or that falls outside the above date ranges is ignored.

Calculating the Bill Start and Bill Stop

Autobill calculates the Bill Start and Bill Stop for the billing period in the same manner as described under **Computing the Bill Start and Bill Stop** on page 4-13, with one exception:

- When using Autobill with Oracle Utilities Meter Data Management, the Bill Start and Bill Stop must fall within the Start Time and Stop Time of the Service Point Account record for the account.

Scenarios

This section outlines several scenarios that illustrate how Autobill works with Oracle Utilities Meter Data Management as described above. In the table below, the following is provided for each scenario:

- **Features**: lists specific settings used for the scenario, such as Billing Mode flag, Bill On Red Date flag, etc.
- **Description**: lists specifics of the bill period used for the scenario, including start and stop times for interval data, service points, billing cycle dates, etc.
- **Expected Results**: lists the computed Bill Start and Bill Stop for each scenario.

Features	Description	Expected Results
Autobill (Set Billing Mode Flag: Account Level) Bill On Read Date: Bill Stop Run Autobill on this date: 01/28/2005	No Bill History Read Date: 01/25/2005 MDM Interval Start: 01/01/2005 Stop: 01/31/2005 23:59:59 Pre-Window: 3 Post-Window: 5	Bill Start: 01/01/2005 Bill Stop: 01/24/2005 23:59:59

Features	Description	Expected Results
Account with Full Day Bill “Yes” on Account History Level. Bill On Read Date: No Run Autobill on this date: 01/28/2005	No Bill History: Read Date: 01/27/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 5 MDM Interval Data Bill Start: 01/01/2005 Bill Stop: 01/27:2005 10:11:12 Full Day Bill Check on Account History Level	Bill Start: 01/01/2005 Bill Stop: 01/26/2005 23:59:59
Account with Scalar Data No Bill History Bill On Read Date: No Run Autobill on this date: 01/31/2005	No Bill History: Billing Cycle Date: 01/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 6 Scalar Data Provided: Start: 01/01/2005 Stop: 01/30/2005 23:59:59	Bill Start: 01/01/2005 Bill Stop: 01/30/2005 23:59:59
Account with Scalar Data And Service Point with Bill Stop Bill On Read Date: No Run Autobill on this date: 01/31/2005	No Bill History: Billing Cycle Date: 01/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 6 Service Point: Start: 01/01/2005 Stop: 01/22/2005 11:11:11 Scalar Data Provided: Start: 01/01/2005 Stop: 01/30/2005 23:59:59	Bill Start: 01/01/2005 Bill Stop: 01/22/2005 11:11:11
Account with Interval Data With Existing Bill History and try to bill on the next cycle Bill On Read Date: Bill Stop Run Autobill on this date: 02/26/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/30/2005 23:59:59 Next Billing Cycle Date: 02/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 8 Service Point: Start: 01/01/2005 Stop: 01/22/2005 Scalar Data Provided: Start: 01/31/2005 Stop: 02/28/2005 23:59:59	Bill Start: 01/31/2005 00:00:00 Bill Stop: 02/24/2005 23:59:59

Features	Description	Expected Results
Account with Scalar Data No Bill History Bill On Read Date: No Run Autobill on this date: 01/31/2005	No Bill History: Billing Cycle Date: 01/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 6 Scalar Data Provided: Start: 01/01/2005 Stop: 01/27/2005 23:59:59	Bill Start: 01/01/2005 Bill Stop: 01/27/2005 23:59:59
Add Account Stop Time Bill On Read Date: No Run Autobill on this date: 01/27/2005	Account: Start Time: 01/01/2005 Stop Time: 01/27/2005 21:19:19 Read Date: 01/27/2005	Bill Start: 01/01/2005 Bill Stop: 01/27/2005 21:19:19
Service Point with Stop Time Bill On Read Date: No Run Autobill on this date: 01/27/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/27/2005 Service Point: Start Time: 01/01/2005 Stop Time: 01/14/2005 23:59:59 Pre-Window: 0 Post-Window: 5	Bill Start: 01/01/2005 Bill Stop: 01/14/2005 23:59:59
Service Point with record switch from one Meter ID (Close the old one) to another Meter ID (create a new one) Bill On Read Date: Bill Stop Run Autobill on this date: 01/27/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/27/2005 Service Point (1): Start Time: 01/01/2005 Stop Time: 01/14/2005 23:59:59 Service Point (2): Start Time: 01/01/2005 Pre-Window: 0 Post-Window: 5	Bill Start: 01/01/2005 Bill Stop: 01/26/2005 23:59:59
2 Service Points with Stop Time in both Bill On Read Date: No Run Autobill on this date: 01/27/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/27/2005 Service Point (1): Start Time: 01/01/2005 Stop Time: 01/14/2005 23:59:59 Service Point (2): Start Time: 01/01/2005 Stop Time: 01/17/2005 23:59:59 Pre-Window: 0 Post-Window: 5	Bill Start: 01/01/2005 Bill Stop: 01/17/2005 23:59:59

Features	Description	Expected Results
Service Point Stop Time and MDM Configuration Stop Time Bill On Read Date: No Run Autobill on this date: 01/27/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/27/2005 Service Point: Start Time: 01/01/2005 Stop Time: 01/14/2005 23:59:59 MDM Configuration: Start Time: 01/01/2005 Stop Time: 01/17/2005 23:59:59 Pre-Window: 0 Post-Window: 3 set up from Account Table.	Bill Start: 01/01/2005 Bill Stop: 01/14/2005 23:59:59
Multiple Interval cuts through the billing period Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 5 MDM Interval Data (1) Bill Start: 01/01/2005 Bill Stop: 01/23:2005 23:59:59 (2) Bill Start: 01/24/2005 Bill Stop: 01/31/2005 23:59:59	Bill Start: 01/01/2005 Bill Stop: 01/24/2005 23:59:59
Multiple Consumption records through the billing period Bill On Read Date: No Run Autobill on this date: 01/30/2005	Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 Read Date: 01/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 5 MDM Consumption Data (1) Bill Start: 01/01/2005 Bill Stop: 01/23:2005 11:11:11 (2) Bill Start: 01/23/2005 11:11:12 Bill Stop: 01/30/2005 23:59:59	Bill Start: 01/01/2005 Bill Stop: 01/30/2005 23:59:59

Features	Description	Expected Results
Multiple Consumption records through the billing period Bill On Read Date: No Run Autobill on this date: 02/25/2005	Previous Bill History: Bill Start: 01/01/2005 Bill Stop: 01/31/2005 23:59:59 Current Bill History: Bill Start: 02/01/2005 Bill Stop: 02/28/2005 Read Date: 02/25/2005 Service Point: Start Time: 01/01/2005 MDM Configuration: Start Time: 01/01/2005 Pre-Window: 3 Post-Window: 5 MDM Consumption Data (1) Bill Start: 02/01/2005 00:15:01 Bill Stop: 02/14/2005 23:59:59 (2) Bill Start: 02/15/2005 Bill Stop: 02/28/2005 23:59:59	Bill Start: 02/01/2005 Bill Stop: 02/28/2005 23:59:59
Add Account Stop Time & Service Point Stop Time Account Stop Time > Service Point Stop Time Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2005	Account: Start Time: 01/01/2005 Stop Time: 01/19/2005 21:19:19 Read Date: 01/25/2005 Service Point: Start Time: 01/01/2005 Stop Time: 01/18/2005 11:11:11	Bill Start: 01/01/2005 Bill Stop: 01/19/2005 21:19:19
Add Account Stop Time & Service Point Stop Time Account Stop Time < Service Point Stop Time Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2005	Account: Start Time: 01/01/2005 Stop Time: 01/14/2005 21:19:19 Read Date: 01/25/2005 Service Point: Start Time: 01/01/2005 Stop Time: 01/18/2005 11:11:11	Bill Start: 01/01/2005 Bill Stop: 01/14/2005 21:19:19
Add Account Stop Time & Service Point Stop Time Account Stop Time = Service Point Stop Time Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2005	Account: Start Time: 01/01/2005 Stop Time: 01/14/2005 21:19:19 Read Date: 01/25/2005 Service Point: Start Time: 01/01/2005 Stop Time: 01/14/2005 21:19:19	Bill Start: 01/01/2005 Bill Stop: 01/14/2005 21:19:19
4 MDM IDs with SP record switch from one Meter ID (Close the two old one before the bill period) to another Meter ID (create two new one) Stop Time at the SP configuration table. Bill On Read Date: Bill Stop Run Autobill on this date: 02/25/2005	Bill History: Bill Start: 02/01/2005 Read Date: 02/25/2005 SP & Meter Configuration (1): Start Time: 01/01/2005 Stop Time: 01/31/2005 23:59:59 SP & Meter Configuration (2): Start Time: 01/01/2005 Stop Time: 01/31/2005 23:59:59 SP & Meter Configuration (3): Start Time: 02/01/2005 SP & Meter Configuration (4): Start Time: 02/01/2005 Pre-Window: 0 Post-Window: 5	Bill Start: 02/01/2005 Bill Stop: 01/24/2005 23:59:59

Features	Description	Expected Results
4 MDM IDs with SP Configuration record switch from one Meter ID (Close the two old one during the bill period) to another Meter ID (create two new one) Stop Time at the SP Configuration table. Bill On Read Date: Bill Stop Run Autobill on this date: 02/25/2005	Bill History: Bill Start: 02/01/2005 Read Date: 02/25/2005 SP & Meter Configuration (1): Start Time: 01/01/2005 Stop Time: 02/14/2005 23:59:59 SP & Meter Configuration (2): Start Time: 01/01/2005 Stop Time: 02/16/2005 23:59:59 SP & Meter Configuration (3): Start Time: 02/15/2005 SP & Meter Configuration (4): Start Time: 02/17/2005 Pre-Window: 0 Post-Window: 5	Bill Start: 02/01/2005 Bill Stop: 01/24/2005 23:59:59
Actual First cut is going to have a date time later then the Account Start Time Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2005	Account: Start Time: 01/03/2005 No Bill History SP (1): Start Time: 01/03/2005 Pre-Window: 0	Generate error because of incomplete data.
MDM configuration Stop Time Bill On Read Date: Bill Stop Run Autobill on this date: 02/25/2005	Account: Start Time: 01/01/2005 Read Dates: 02/25/2005 SP (1): Start Time: 01/01/2005 MDM Configuration (1): Start Time: 01/01/2005 Stop Time: 01/15/2005 23:59:59 Billed: “Y” MDM Configuration (2): Start Time: 02/01/2005 Billed: “Y” Pre-Window: 0 Post-Window: 5	Bill Start: 02/01/2005 Bill Stop: 02/24/2005 23:59:59
Bill the account service point 2 because account service point 1 was stopped Bill On Read Date: Bill Stop Run Autobill on this date: 01/25/2006	Account: Start Time: 01/03/2005 BH: Bill Start: 01/01/2006 Bill Stop: 01/31/2006 23:59:59 Read Dates: 01/25/2006 SP (1): Start Time: 01/03/2005 Stop Time: 06/03/2005 23:59:59 MDM Configuration (1): Start Time: 01/03/2005 Billed: “Y” Provided: Interval Data from 01/03/2005 – 06/03/2005 23:5959 SP (2): Start Time: 01/03/2005 MDM Configuration (2): Start Time: 01/03/2005 Billed: “Y” Provided: Interval Data from 01/03/2005 – 01/31/2006 23:5959 Pre-Window: 0 Post-Window: 5	Bill Start: 01/01/2006 Bill Stop: 01/24/2006 23:59:59

Chapter 5

Setting Up Oracle Utilities Billing Component Database Records

This chapter describes how to set up database records used by Oracle Utilities Billing Component, including:

- **Setting Up Oracle Utilities Billing Component Tables**
- **Using Work Queues with Oracle Utilities Billing Component**

You set up database records in the Oracle Utilities Data Repository using either Data Manager or Data Navigator.

Setting Up Oracle Utilities Billing Component Tables

Oracle Utilities Billing Component requires that records be set up in the following tables in the Oracle Utilities Data Repository:

- **Account History**
- **Account Note Types**
- **Bill Codes**
- **Bill Determinant Use**
- **Billing Cycle Dates**
- **Billing Cycle**
- **Inter-Schedule Mapping**
- **Rate Code**
- **Rate Code History**
- **Rebill Reason**

Account History

Records in the Account History table store relationships between accounts and billing cycles (defined in the **Billing Cycle** table) and Rules Language contracts. These records help Oracle Utilities Billing Component determine the correct read date for an account. Records in this table include the following information:

- **Account**: The account associated with the relationship.
- **Effective Date**: The date on which the relationships defined in the record are effective.
- **Contract**: The Rules Language contract associated with the account.
- **Billing Cycle**: The Billing Cycle (from the **Billing Cycle** table) associated with the account.
- **Bi-Monthly**: A flag that indicates if the account is to be billed b-monthly.
- **Full Day Bill**: A flag that indicates if the account should be billed using Full day bill. See **Full Day Bill** on page 7-12 for more information.
- **Billing Cycle - Governing**: The governing Billing Cycle for the account. See **Effective and Governing Dates** on page 4-4 for more information.
- **Bill on Read Date**: A flag that indicates if the account is to be billed on or before the read date. See **Bill On Read Date** on page 4-5 for more information.

Account Note Types

Account Notes are primarily used to alert billing analysts to an actual or potential problem that has occurred during the billing process. An account note can be recorded by the software as it attempts to process the account, or by a user examining the account. The note can stop the bill calculations or merely provide information, as you specify. Records in this table include the following information:

- **Code:** A unique code (up to 64 characters) identifying the account note type.
- **Name:** A descriptive name (up to 64 characters) for the note.
- **Stop Billing:** Select from four options:
    - **Automatic Only -** Note prevents billing for accounts on “Fully Automatic” billing mode.

- **Both -** Note prevents billing for accounts on either "Fully Automatic" or "Approval Required" mode.
- **Manual Only -** Note prevents billing for accounts on "Approval Required" mode.
- **No -** Note does not prevent billing (Account Note will be informational only).

  Note: When using work queues, exceptions are written to the Work Queue Open Item table instead of the Account Notes table, eliminating the need to create Account Note Type records. To enable work queues, include the **USE_WORK_Q_FOR_ACCOUNT_NOTES = 1** parameter in the LODESTAR.CFG file.

Bill Codes

Records in the Bill Codes table represent types of bills. This table is used as a lookup in the Bill History table. Records in this table include the following information:

- **Code**: A unique code for the bill code.
- **Name**: The name of the bill code.

Bill Determinant Use

Records in the Bill Determinant Use table indicate that a specific Bill Determinant (from the Bill Determinant table) is used by an account. Records in this table include the following information:

- **Account**: The account which uses the bill determinant specified in the record.
- **Bill Determinant**: The bill determinant used by the account (from the Bill Determinant table).
- **Start Time**: The start time for the relationship between the account and the bill determinant.
- **Stop Time**: *Optional.* The stop time for the relationship between the account and the bill determinant.

Billing Cycle

Records in the Billing Cycle table identify the billing cycles used by operating companies/ jurisdictions. The actual dates for each billing cycle are stored in the Billing Cycle Date records. Records in this table include the following information:

- **Operating Company Code:** The operating company that uses this billing cycle.
- **Jurisdiction:** The jurisdiction that uses this billing cycle is applied.
- **Code:** A unique code (up to 64 characters) identifying the billing cycle.

Billing Cycle Dates

Records in the Billing Cycle Dates table are child records of Billing Cycles, and store the read date for each bill month in a billing cycle. Records in this table include the following information:

- **Billing Cycle**: The billing cycle to which the dates apply. This includes the operating company and jurisdiction of the billing cycle.
- **Read Date:** The meter read date, in the MM/DD/YYYY format. This field is used by the system to match an account's Bill History or Meter Value records with the correct bill period.
- **Bill Month:** The bill month, in the MM/YYYY format.

Inter-Schedule Mapping

Records in the Inter-Schedule Mapping table define “equivalencies” between rate codes in different rate schedules. This makes it possible to substitute the “best” rate code for a customer when comparing different rate schedules, such as when applying Oracle Utilities Rate Management’s Customer Impact Analysis or Oracle Utilities Billing Component’s Trial Bill Calculation. See **Appendix B**: **Mapping Rate Codes Between Rate Schedules** for more information.

Records in this table include the following information:

- **Rate Code**: The rate code to which the other rate code is mapped.
- **Mapped Rate Code**: The rate code to be mapped to the rate code defined in the **Rate Code** field.

Rate Code

Records in the Rate Code table represent rate codes (associated to rate schedules) that define tariffs or rates structures. Records in this table include the following information:

- **Rate Schedule**: The rate schedule associated with the rate code.
- **Code**: A unique code for the rate code.
- **Note**: An optional note for the rate code.
- **Billing Mode Flag**: An optional flag that indicates the billing mode (Fully Automatic, Approval Required, or Manual Start) for bills calculated using this rate code. See **Billing Modes** on page 7-8 for more information.
- **Print Detail**: An optional flag that indicates the print detail (None, Normal Detail, or All Detail) for bills calculated using this rate code. See **Print Detail** on page 7-10 for more information.
- **Full Day Bill**: An optional flag that indicates if accounts billed using this rate code should be billed using Full day bill. See **Full Day Bill** on page 7-12 for more information.

Rate Code History

Records in the Rate Code History table represent relationships between accounts and rate codes (from the Rate Code table). Oracle Utilities Billing Component and Oracle Utilities Rate Management use the records in this table to determine which rate schedules to use when processing billing or analysis calculations for an account. Records in this table include the following information:

- **Account**: The account associated with the rate code.
- **Start Time**: The start time for the relationship between the account and the rate code.
- **Rate Code**: The rate code associated with the account. This includes the operating company, jurisdiction, rate schedule code, and rate code.
- **Stop Time**: *Optional.* The stop time for the relationship between the account and the rate code.
- **Note**: An optional note for the rate code.

Rebill Reason

Records in the Rebill Reason table represent specific reasons why an account might need to be rebilled (using Bill Correction). Records in this table include the following information:

- **Code**: A unique code for the rebill reason.
- **Rebill Reason Name**: The name of the rebill reason.

Using Work Queues with Oracle Utilities Billing Component

Oracle Utilities Billing Component can be configured to automatically send exceptions and work queue items to the Work Queues application. Oracle Utilities Billing Component normally writes exceptions to the Account Notes table. When using the work queues alongside Oracle Utilities Billing Component, exceptions can be written to the Work Queue Open Item table instead of the Account Notes table.

Configuring Oracle Utilities Billing Component to use the Work Queues application involves the following steps:

- **Setting Up Work Queue Type Records**
- **Enabling Work Queues**
- **Rules Language Configuration**

Setting Up Work Queue Type Records

This step involves creating Work Queue Type records for the types of exceptions or errors that you want to record with work queue items. Work Queue Type records that match the Account Note Type records used in previous versions of Oracle Utilities Billing Component are delivered with Oracle Utilities Billing Component and are also pre-configured in the Work Queue and Work Queue Types table, and include the following:

- **ABORT**: ABORT statement executed during bill calculation.
- **BILLED**: Account had been billed.
- **EXCEPTION**: Data missing on day after read date.
- **HOLD**: User requested hold.
- **INFO**: User entered.
- **SYSTEM REJECTED**: Failure during billing - validation or calculation.
- **USER REJECTED**: User rejected during manual review.
- **USERCALC**: User computed bill via Current Bill.
- **WARNING**: Warning statement executed during bill calculation.

If your Oracle Utilities Billing Component implementation requires additional work queue types, you must create the appropriate Work Queue Type records for them. Work queue items based on these additional (non-Account Note) work queue types must be created via the Oracle Utilities Rules Language configuration used by Oracle Utilities Billing Component.

When creating Work Queue Types for use with Oracle Utilities Billing Component, the **Stop Billing** flag should be set appropriately, based on your desired behavior. This flag provides four options:

- **Automatic Only -** Work queue items stop billing for accounts on “Fully Automatic” billing mode.
- **Both -** Work queue items stop billing for accounts on either “Fully Automatic” or “Approval Required” mode.
- **Manual Only -** Work queue items stop billing for accounts on “Approval Required” mode.
- **No -** Do not stop billing (work queue items will be informational only).

See **Setting Up Work Queue Types** on page 5-12 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about creating Work Queue Type records.

Enabling Work Queues

To enable the Work Queues application to work with Oracle Utilities Billing Component, include the **USE_WORK_Q_FOR_ACCOUNT_NOTES = 1** parameter in the LODESTAR.CFG file. See **LODESTAR.CFG** on page 2-2 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information.

Rules Language Configuration

Once the Work Queues functionality has been enabled, Oracle Utilities Billing Component will create work queue items based on Account Notes automatically. Work queue items based on other (non-Account Note) work queue types must be created via the Oracle Utilities Rules Language configuration used by Oracle Utilities Billing Component.

When creating Work Queue items for use with Oracle Utilities Billing Component, the following are required in order for the work queue items to prevent billing (based on the value of the **Stop Billing** flag):

- The **Account ID** column must be populated with the account for which the exception occurred
- The **Read Date** column must be populated with the read date for the billing period being processed.

See **Using the Oracle Utilities Rules Language** on page 5-22 in the *Oracle Utilities Energy Information Platform Configuration Guide* for more information about configuring the Oracle Utilities Rules Language to create work queue items.

Chapter 6

Setting Up and Applying Overrides and Special Events

Overrides enable you to apply alternative charges for customer energy use during “special events” or other user-specified periods. This chapter explains how to set up “Overrides” for accounts in the Oracle Utilities Data Repository, including:

- **What are Overrides?**
- **Setting Up Overrides in the Oracle Utilities Data Repository**
- **Approving Overrides Prior to Billing (Override Ready Table)**

What are Overrides?

An “override” is a mechanism for applying alternate charges to a bill. There are two basic types of overrides:

- Discounts or other modifications that are applied routinely to the customer’s bill, typically over an extended period. For example, a customer eligible for an “economic development discount” might receive a 10% discount off the total monthly bill for a year.
- Penalties or discounts that are triggered by the occurrence of a “special event,” such as a maintenance or standby period. For example, a customer might agree to curtail their energy use for a few hours or days during times of high system-wide demand, in return for a specified discount.

The specific charges based on the overrides must be defined in a rate form using Rules Language statements. The circumstances for invoking the override for an account, the time period over which it is applied, and any time-dependent variables are specified in the Override tables in the Oracle Utilities Data Repository. These tables also let you enable a validation feature for the application of the special event overrides, called an “Account Ready” flag.

There are five Override tables in the base system:

- **Override Table:** This table assigns a unique code to identify each override in the system.
- **Override History Table:** An entry in this table applies an override to an account. It provides the specific information Oracle Utilities Billing Component needs to apply the override to the calculations for the customer’s bills, such as the time period that the override is in effect and values for any variables in the rate form.
- **Name Override History Table:** An entry in this table applies an override to an account *at the meter level*. For example, an account may have several separately metered locations, each subject to different overrides. Otherwise, this table is the same as the Override History Table.
- **Override Use Table:** *Optional.* An entry in this table triggers the “Override Ready” requirement. This is used only for special event overrides that are applied sporadically and need verification (see below).
- **Override Ready Table:** If an account has an entry in the Account Override Use Table, a billing analyst must verify that the information in the Override History or Name Override History Table is up to date for the current bill period. Afterwards, the analyst must manually set the “Ready” flag in the Override Ready Table before Oracle Utilities Billing Component will calculate the account’s bill.

You can create a Special Event report that lists all overrides that have been applied to an account or an account’s channel group over a user-specified period See **Chapter 11**: **Other Reports** in the *Oracle Utilities Billing Component User’s Guide* or **Chapter 10**: **Billing Reports** in the *Oracle Utilities Billing Component Web User’s Guide* for more information.

Setting Up Overrides in the Oracle Utilities Data Repository

This section describes how to set up overrides for use with Oracle Utilities Billing Component, including:

- **Defining Override Codes (Override Table)**
- **Applying an Override to an Account or Channel Group**
- **Enabling "Account Ready" Validation for Special Event Overrides**

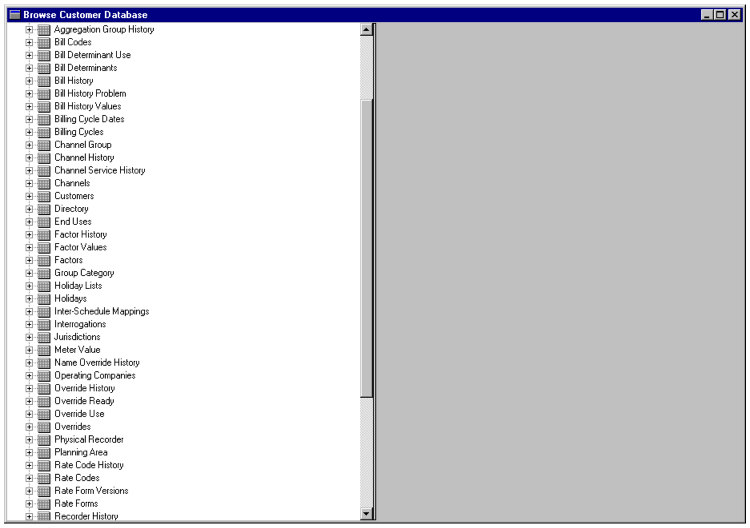
Defining Override Codes (Override Table)

You must define an override in this table before you can apply it elsewhere in the system.

How to enter an override lookup code using the Data Manager Browser:

1. In Data Manager, select **Browse->Customer Database**.

   The Browser appears.

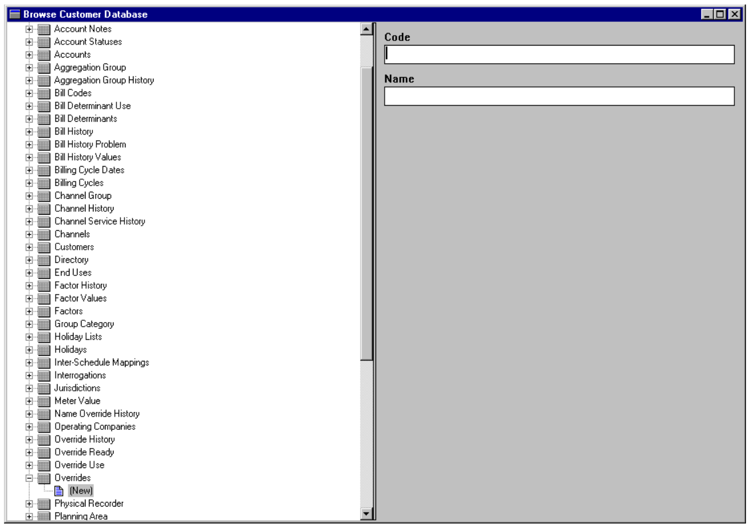


2. Click on the **Overrides** table icon.

   A list of all currently defined override codes appears in the right pane.

3. Select **Records->Insert New,** or click the *right* mouse button.

   A blank data entry form appears.



4. Complete the form.

   Code: Enter a unique code (up to 64 characters) to identify the override. It will be used in the rate form scripts to apply the override.

   Name: Enter a unique name for the override. This name will appear in the Oracle Utilities Billing Component user interface to identify the override.

5. Select **Edit->Add** to save the information to the database.

   The new override code appears in the tree pane on the left.

6. Select **Records->Refresh** or **Records->Refresh All** to update the displays with the new information.

Applying an Override to an Account or Channel Group

To include an override in an account's bill, you must make an entry in the Override History or Name Override History Table, depending upon whether the override is applied at the account or meter level, respectively. For long-term overrides, such as economic development discounts, you might make one entry that covers a number of months or even years. For a "special event" override, you might have to make a number of entries for each bill period.

How to apply an override to an account:

1. Select **File->Browse Customer/Accounts**.

   The **View Customer/Account(s) Input** dialog box appears.

2. Enter the ID of the desired account, and click **OK**.

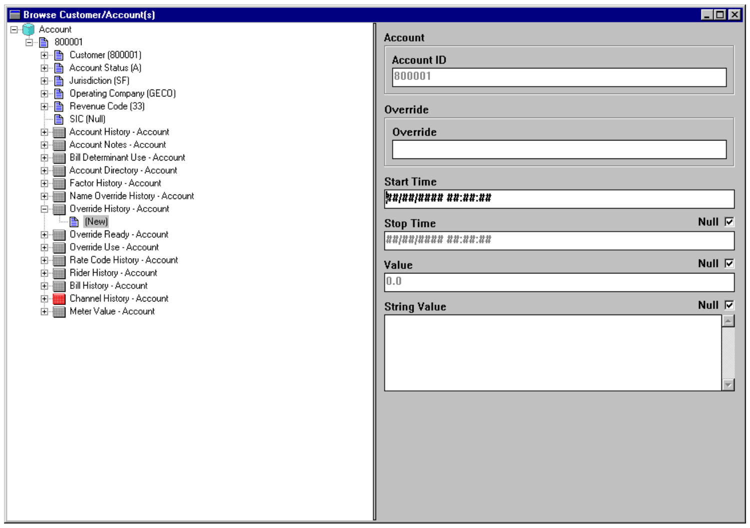
   The Browser appears. The records displayed in the tree pane are limited to those that apply to the selected account.

3. Click on the **Override History** or **Name Override History** table icon in the tree pane, depending upon whether the override is applied at the account or meter level.

   A list of overrides that have been applied to this account in the past appears in the right pane.

4. Select **Records->Insert New,** or click the *right* mouse button.

   A blank data entry form appears.



Account
- 800001
  - Customer (800001)
  - Account Status (A)
  - Jurisdiction (SF)
  - Operating Company (GECO)
  - Revenue Code (33)
  - SIC (Null)
  - Account History - Account
  - Account Notes - Account
  - Bill Determinant Use - Account
  - Account Directory - Account
  - Factor History - Account
  - Name Override History - Account
  - Override History - Account
    - (New)
  - Override Ready - Account
  - Override Use - Account
  - Rate Code History - Account
  - Rider History - Account
  - Bill History - Account
  - Channel History - Account
  - Meter Value - Account

Account

Account ID: 800001

Override

Override:

Start Time: ##/##/#### ##:##:##

Stop Time: ##/##/#### ##:##:## Null ✓

Value: 0.0 Null ✓

String Value: Null ✓

5. Complete the form:

   Override: Click on the word "Override". A list of available overrides appears (if the override you want does not appear, it has not been defined in the Override Table). Select the desired code from the list that appears, and press ENTER or click **OK**.

   Name: *For Name Override History only.* Enter the name of the channel group, CIS account, or "recorder-ID,channel" to which this override applies.

   Start Time: Enter the beginning of the period over which the override applies.

   Stop Time: Enter the end of the period over which the override applies. If the override has no stop date, the stop date is unknown, or the override applies indefinitely, check the Null checkbox.

   Value: *Optional.* Enter a value that expresses the magnitude of the override. Oracle Utilities Billing Component will use this value for a numeric variable in the rate form (see the *Oracle Utilities Rules Language User's Guide*). If the value refers to a percentage, express it as a decimal fraction. For example, enter 0.1 to indicate a 10% discount.

   String value: *Optional.* Enter a value for a variable in the rate form script that requires a text string (see the *Oracle Utilities Rules Language User's Guide*). For example, if you wanted to apply a rider for just the period defined by the start- and stop-times above, enter the name of the rider. If you wanted to apply a factor for just that period, enter the factor's name.

   Note: *Optional - for Account Override History only*. Enter descriptive information if desired.

6. Select **Edit->Add** to save the information to the database.

Enabling "Account Ready" Validation for Special Event Overrides

The following procedure is necessary only if you want to require that a billing analyst manually validate the override for the account before billing. This feature is typically used for overrides that are applied sporadically, such as in response to a "Special Event." If this record is in place, the account's bill cannot be issued until the analyst sets the account override ready flag, regardless of whether the override is actually applied to the account during the billing period.

> **Note**: Instructions for setting the account override ready flag can be found in the next section of this chapter.

How to Enable Account Ready Validation for Special Event Overrides

1. Select **File->Browse Customer/Accounts**.

   The **View Customer/Account(s) Input** dialog box appears.

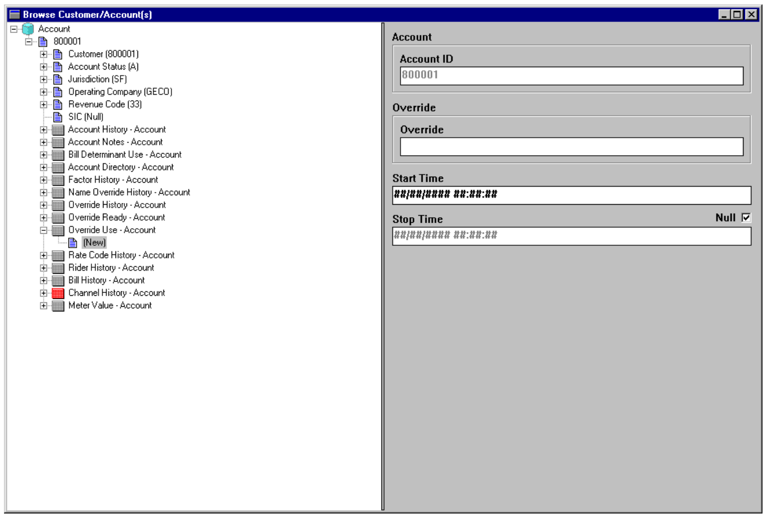
2. Enter the ID of the desired account, and click **OK**.

   The Browser appears. The records displayed in the tree pane are limited to those that apply to the selected account.

3. Click on the **Override Use** table icon in the tree pane.

4. Select **Records->Insert New** or click the *right* mouse button.

   A blank data entry form appears in the right pane.



Account

Account ID: 800001

Override

Override:

Start Time: ##/##/#### ##:##:##

Stop Time: ##/##/#### ##:##:## Null ✓

5. Complete the form:

   Override: Click on the word “Override”. A list of available overrides appears. Select the desired code from the list, and press ENTER or click **OK**.

   Start Time: The date and time that the account qualified for the override.

   Stop Time: The date and time that the override is scheduled to end. If an end date is not known, or if the override applies indefinitely, select Null.

6. Select **Edit->Add** to save the information to the database.

7. Select **Records->Refresh** or **Records->Refresh All** to update the displays with the new information.

Approving Overrides Prior to Billing (Override Ready Table)

If there is an active entry in the Override Use Table for an account, a billing analyst must note all occurrences of the override for that account during the bill period in the Override History or Name Override History Table, and specify that the information is accurate and up to date by setting the Ready flag in the Override Ready Table. *Even if the override did not apply during the bill period, the billing analyst must still set the Ready flag before the bill can be issued.* This ensures that no potentially-applicable overrides are overlooked. In addition, the billing analyst must perform this action within *n* days before the bill is issued, where *n* is a number defined in the **Special Events Pre Window** field in the **Default Billing Options** dialog box. See **Chapter 7***:* **Setting Global Defaults and Applying Settings to Accounts, Rate Codes, and Rate Schedules** for more information.

How to set the "Ready" flag for an account/override:

1. Select **File->Browse Customer/Accounts**.

   The **View Customer/Account(s) Input** dialog box appears.

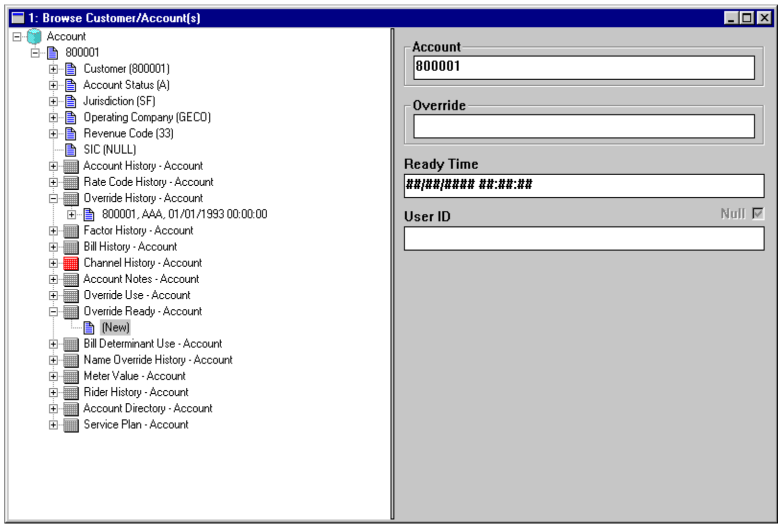
2. Enter the ID of the desired account, and click **OK**.

   The Browser appears. The records displayed in the tree pane are limited to those that apply to the selected account.

3. Click on the **Override Ready** table icon in the tree pane.

4. Select **Records->Insert New,** or click the *right* mouse button.

   A blank data entry form appears.



1: Browse Customer/Account(s)

```
Account
  800001
    Customer (800001)
    Account Status (A)
    Jurisdiction (SF)
    Operating Company (GECO)
    Revenue Code (33)
    SIC (NULL)
    Account History - Account
    Rate Code History - Account
    Override History - Account
      800001, AAA, 01/01/1993 00:00:00
    Factor History - Account
    Bill History - Account
    Channel History - Account
    Account Notes - Account
    Override Use - Account
    Override Ready - Account
      (New)
    Bill Determinant Use - Account
    Name Override History - Account
    Meter Value - Account
    Rider History - Account
    Account Directory - Account
    Service Plan - Account
```

Account: 800001

Override:

Ready Time: ##/##/#### ##:##:##

User ID: Null ✓

5. Complete the form:

   Override: Click on the word "Override". A list of available overrides appears. Select the desired code from the list, and press ENTER or click **OK**.

   Ready Time: Enter the date and time that you verified that all instances of this override for this account for this bill period have been entered correctly.

   Note: Typically, the Ready Time will be the current date and time.

   User ID: The user ID of the user creating the record. This is filled in automatically, and cannot be edited.

6. Select **Edit->Add** to save the information to the database.

7. Select **Records->Refresh** or **Records->Refresh All** to update the display with the new information.

Chapter 7

Setting Global Defaults and Applying Settings to Accounts, Rate Codes, and Rate Schedules

To accommodate the widely varying bill requirements of individual customers and at the same time maximize efficiency in the billing process, Oracle Utilities Billing Component gives you a great deal of flexibility for tailoring the system to both individual accounts and groups of accounts. For example, you can specify how bill calculations are initiated and approved, how customer usage data is processed, and the level of detail included in a bill. You can apply the options at multiple levels of the billing hierarchy. For example, you can specify options for an individual account, a selected rate schedule, or a rate code.

You can set the options in a number of places in the system—by checking global options in the Oracle Utilities Billing Component user interface, by inserting "flags" in records in the Oracle Utilities Data Repository tables, and by inserting Rules Language statements in rate forms.

This chapter describes each of the billing options available to you, and the various methods for setting them, including:

- **Default Billing Options**
- **Billing Modes**
- **Print Detail**
- **Full Day Bill**
- **Eligibility Windows**
- **Setting Up Summary Bills for Customers**
- **Specifying Usage Data Required to Bill an Account**
- **Applying Tariff Riders to Accounts**

Default Billing Options

The **Default Billing Options** dialog box lets you specify default global options that apply to all accounts in the system. These options apply to all billing modules and to all Oracle Utilities Billing Component users (unlike the user-specific **Tools-›Options** preferences described in **Chapter Two** of the *Oracle Utilities Billing Component User's Guide*). However, you can override any of these settings for specific rate schedules, rate codes, or accounts in the Oracle Utilities Data Repository, or for individual jobs. Available Billing options include:

- **Billing Options**
- **Billing Rules**
- **Report Options**
- **Summary Options**
- **Multiple Bill History Records**
- **Check Options**

How to set a global billing mode:

1. Select **CIS Billing (or Analysis)-›Billing Options**.

   The **Default Billing Options** dialog box appears.
2. Set your options on each tab as desired.
3. When you have completed the tabs as desired, you have the following choices for applying the options:
   - **OK:** Saves the new settings as your defaults, replacing the previous settings.
   - **Apply:** The new settings apply to all of your work for remainder of your current session. The settings will revert to the original defaults when you exit the application.

Billing Options

The Billing Options tab specifies a number of default values and preferences used when performing billing calculations.

Default Billing Mode: Use this option to specify the mode that Oracle Utilities Billing Component applies to accounts that have not otherwise been assigned a billing mode. If you select **Automatic Billing**, Oracle Utilities Billing Component calculates the bills, updates the Oracle Utilities Data Repository, and issues transaction records—all without review and approval by a user. If you select **Approval Required**, Oracle Utilities Billing Component puts the results of its bill calculations in a bill report which you must approve before the software issues any transaction records or updates the bill history record in the Oracle Utilities Data Repository).

Note: Any billing mode flags set for individual rate schedules, rate codes, or accounts in the Oracle Utilities Data Repository records automatically override this setting.

Default Eligibility Pre- and Post-Windows: Specify the number of days before an account's scheduled read date (the Pre-Window) and the number of days after an account's scheduled read date (the Post-Window) that defines the "eligibility" window. You can supply any value from 0 to 28 for either. In the **Automatic Billing** and **Approval Required** modes, Oracle Utilities Billing Component uses this window to identify "eligible" accounts. That is, if today's date falls within this window, Oracle Utilities Billing Component begins scanning for the data required to bill the account and continues scanning until the data is available or the window for the account passes today.

Note: If you set the Post Window to 0, the data must be available by the end of the read date, or it is considered missing. You can override the eligibility window setting for individual accounts and recorder types.

Note: The Default Pre- and Post-Windows should be set the same or larger than the largest Pre- and Post-Window for any single account.

Special Event Pre Window: Specify the number of days prior to the billing date within which a Billing Analyst must approve an account's Special Events information in order for the account to be billed. You can enter any value from 0 to 28.

This feature is designed to ensure that all Special Events are included in a bill. For example, suppose that the Billing Analyst approved an account's Special Event information on schedule. However, for some reason the account's meter could not be read until seven days after the scheduled date. That would mean there would be seven days during which a Special Event could occur but not be included in the current bill. However, if you set the Special Event Pre Window to some number less than 7, Oracle Utilities Billing Component would flag the account as "ineligible" and the billing analyst would have to update the special events information.

Automatic Repeat Frequency: The default amount of time between a completed Automatic Billing cycle and the next. Set this number in minutes, from 0 (do not repeat) to 1440 (24 hours later).

Note: The setting here does not actually define the repeat frequency; it simply specifies the default. The repeat frequency is set on the Automatic BIlling Parameters dialog box.

Billing Rules

The Billing Rules tab specifies a number of default rules and settings used when performing billing calculations.

Effective Date is: The date used to determine Rate Form versions, factor values and the default for determining the season. There are four options:

- **Bill Stop**
- **Bill Start**
- **1st of Bill Month**
- **Scheduled Read Date**

See **General Billing Rules** on page 4-4 of the *Oracle Utilities Billing Component Installation and Configuration Guide* for more information.

Allow Saves without SAVE TO CIS statement: Normally output records are created as part of a bill calculation. These records are used with non-Oracle Utilities Billing Component systems such as printing, accounts receivable, and so on. Oracle Utilities Billing Component verifies that at least one output record has been created before saving any data to the Oracle Utilities Data Repository. Check this box to turn this verification off, then you can save to the database without creating an output record.

You must Approve a bill to save it to the Oracle Utilities Data Repository. When you Approve the bill, data is saved and the Bill Time (BILLTIME) column in the Bill History is set.

Retrieve all Account Determinants: Normally, to improve performance, Oracle Utilities Billing Component loads only the determinants that are explicitly used in the Rate Schedule. However, if you are using indirection (the @ operator) to retrieve determinant values you need to turn this option on. Oracle Utilities Billing Component then loads all determinants in the account's Bill History records.

Use 2.10 and before Account Note generation in Automatic Billing and Approval Required: Check this checkbox to generate Account Notes/Work Queue Items in Automatic and Approval Required billing as was in version 2.10 and earlier versions of Oracle Utilities Billing Component. The primary difference between pre-2.10 and post-2.10 is that for 2.10 and earlier, Account Notes/Work Queue Items are generated on the day after the scheduled read date, while for post-2.10 they are generated the day after the end of the postwindow from the scheduled read date. See **Billing Account Note/Work Queue Item Generation** on page 4-11 in the *Oracle Utilities Billing Component Installation and Configuration Guide* for more information.

Note: When using Work Queues, exceptions are written to the Work Queue Open Item table instead of the Account Notes table. To enable the Work Queues application, include the **USE_WORK_Q_FOR_ACCOUNT_NOTES = 1** parameter in the LODESTAR.CFG file.

Report Options

The Report Options tab specifies a number of default values used with printing reports.

Print Detail: This option determines the level of detail included in the bill reports, transaction records, and bill faxes or E-mail. Select one of the following:

- **None:** Effective revenue only.
- **Normal:** Effective revenue, plus any charges assigned to a revenue identifier and any values assigned to a Report Statement in the applicable rate form.
- **All:** Effective revenue, plus revenue identifiers, plus the values associated with the Rules Language Label and Report statements in the rate form.

Note: You can override this setting for selected accounts, rate codes, or rate schedules using the Billing Mode Flags in the corresponding records in the Oracle Utilities Data Repository.

Show Bill History Problems/Resolutions in Bill Reports: If checked, this option specifies that any problems or resolutions that a Billing Analyst has entered into the Bill History Problem record for the current bill period appear near the top of the bill report (in the header section, just after the customer and account information). This option applies to all bill reports produced by Oracle Utilities Billing Component.

Summary Options

The Summary Options tab specifies how summary customers and accounts are processed. See **Summary Bill Calculation** on page 4-20 in the *Oracle Utilities Billing Component Installation and Configuration Guide* for more information.

Compute Multiple Rates per Account: This option specifies whether all effective rates, or only the most recent rate is used for billing. The options available include:

- **No:** Oracle Utilities Billing Component calculates each account’s bill using just the most recent rate specified in the Rate Code History Table for that account.
- **Throughout Bill Period:** Compute multiple rates using all rates in effect for that account throughout the Bill Period, as specified in the Account Rate Code History Table. When computing multiple rates, Oracle Utilities Billing Component executes the rates in reverse order starting with the record with the most recent Start Time.

Note: If you use either of the first two options, be sure that all inactive rate codes have a stop time in the Account Rate Code History Table; otherwise they will be included in the calculations.

- **On Effective Date:** Compute multiple rates using all rates in effect for that account on the Effective Date, as specified in the Account Rate Code History Table.

Enable Customer SUMMARY Rate Schedules: If this is checked, Oracle Utilities Billing Component can process Summary rate schedules.

Multiple Bill History Records

The Multiple Bill History Records tab specifies how customers/accounts with multiple Bill History records per bill period and/or read date are handled. See **Multiple Bill History Records per Bill Period** on page 4-29 in the *Oracle Utilities Billing Component Installation and Configuration Guide* for more information.

Allow Multiple Bill History Records per Bill Period/Read Date: Normally, Oracle Utilities Billing Component uses the billing cycle's scheduled meter read dates to match an account's Bill History records with the correct bill periods. However, this may not suffice in all cases, either because the metered data does not come in from the field on a reliable schedule, or because the metering and billing schedule at your company is more complex. This tab enables you to choose the action that Oracle Utilities Billing Component takes when it encounters two or more account Bill History records associated with the same read date. The option you choose depends upon the metering and billing policies of your company. (This setting applies to Automatic Billing, Approval Required, and Current/Final Bill.)

- **No:** Oracle Utilities Billing Component rejects all Bill History records for the specified account that have the same read date. It does not process the bills, and issues an error message instead.
- **Bill Individually - Allow Out of Order Bill Periods:** This option is designed primarily for situations in which metered data comes in from the field irregularly—for example, readings for bill periods arrive out of sequence, and/or scheduled read dates are missed altogether. If you select this option, Oracle Utilities Billing Component treats any account Bill History records that have the same read date as valid records, and processes them as separate and complete bills. It retrieves and bills the previously unbilled Bill History records for the account (Bill Time value equals NULL), in sequence, from oldest to most recent. To determine the bill period that the Bill History record actually belongs to, it looks at the Bill History record's Start Date/Time for the metered data and processes the bill accordingly. If the applicable rate form includes any SAVE statements, the computed values are saved to the Bill History record for the correct bill period.
- **Bill Individually - Monthly Merge Historical Values:** This option is designed primarily for companies whose bill calculations require determinant values for sub-periods within the current bill period (e.g., weekly kWh), as well as values for entire historical bill periods (e.g., maximum kW for the past 12 bill months). In this case, multiple sub-period records (e.g., records with different start- and stop-dates) may be associated with a single scheduled read date. If you select this option, Oracle Utilities Billing Component assumes that multiple bill history records with the same read date for one account are in fact sub-period records. To perform the calculations (depending upon the rate schedule), Oracle Utilities Billing Component first aggregates historical sub-period values and start- and stop-dates to find values for entire historical bill periods. It then bills each sub-period for the bill period (same scheduled read date), from oldest through most recent.
- **Aggregate to Read Date and Bill as One:** This option is designed primarily for companies that take multiple meter readings for an account within a single bill period (i.e., partial reads), but bills the account using values for the entire bill period. In this case, two or more Bill History records are allowed per read date. If you select this option, Oracle Utilities Billing Component aggregates all determinant values and dates to the read date (in other words, records with the same read date are combined into one). If the applicable rate form includes any SAVE statements for the sub-periods, it saves them to the latest record with the current read date.

- **Aggregate to Read Date and Bill as One (Oracle Utilities Rate Management Style):** This option is used only for testing, and should not be selected for billing in a production mode. It is similar to the option described above, except that the process is oriented to bill months. That is, two or more Bill History records with the same bill month are combined into one record for the entire bill month.

Check Options

The Check Options specifies how certain checks are performed before processing calculations. In some systems, some of these checks are unnecessary. Use these options to turn off the unneeded checks, to improve performance.

Do not use Interval Data: Turn off all support for Interval Data during calculations.

Do not use Interval Data to Compute Start/Stop: Don't use interval data cuts to compute the bill start and stop, and do not verify that all the account's interval data is ready. If neither Interval Data check is turned off and Multiple Bill History Records are allowed, only the verification is performed.

Do not check for Interval Data gaps: Turn off checking for gaps between cuts. The rate schedule should then check for missing data (status code '9') to validate the data. If this is not checked, a Warning will be reported if there are gaps between the interval data cuts.

Do not check Account Notes: Don't check for Account Notes that stop billing in Automatic Billing or Approval Required. Do not write Account Notes from Automatic Billing or Approval Required.

Do not check Override Ready: Don't check that Special Events or Overrides are Ready.

Do not check Account's Meter Values: Don't verify that account's scalar data is in the Meter Value Table.

Do not check Rate Code Flags: Do not use the BILLINGMODEFLAG, PRINTDETAIL or FULLDAYBILL values in the RATECODE Table. If this is checked, it also turns off the same flags in the Rate Forms (RATEFORM) Table.

Do not check Rate Schedule Flags: Do not use the BILLINGMODEFLAG, PRINTDETAIL or FULLDAYBILL values in the RATEFORM Table, independent of the RATECODE Table.

Do not display "Reason for Change" dialog when updating the Bill History [Value] records: Don't display the Reason for Change dialog after updating Bill History or Bill History Value records.

Do not check Bill Determinant Use: Don't check the Bill Determinants' Use table.

Use MDM schema: When this option is checked, Oracle Utilities Billing Component checks the Oracle Utilities Meter Data Management (MDM) tables, specifically the Meter Data Channel Cut table, for interval data when billing accounts.

Checking this option also automatically checks the following other options:

- Do not check for Interval Data Gaps
- Do not check Account's Meter Values

It also disables the following options:

- Do not use Interval Data
- Do not use Interval Data to compute Start/Stop

Use MDM check data: When this option is checked, usage in the Oracle Utilities Meter Data Management tables is validated when Oracle Utilities Billing Component checks for data.

Each of these checks requires at least one query to a database so turning one off eliminates its database query and saves time. The most time consuming check is using Interval Data to Compute Start/Stop, followed by checks for Account Notes, Account's Meter Values, Override Ready and Bill Determinant Use. To determine what you can turn off, look at the rate schedules. If no rate schedule, rider or contract uses Overrides, for example, turn off the Override Ready check. Likewise, if there your system doesn’t use interval data, turn off the Interval Data checks.

Billing Modes

This section describes the different billing modes and how to set them.

- **What Billing Modes are Available?**
- **Billing Mode Flags**

What Billing Modes are Available?

There are three ways to process bills in Oracle Utilities Billing Component:

- **Fully Automatic:** Oracle Utilities Billing Component finds all eligible accounts based on billing cycle dates, calculates charges, saves data for successful bills to the Oracle Utilities databases as specified in the rate schedule, and issues transaction records to the output file — all without user intervention.
- **Approval Required:** When you activate the **Approval Required** module, Oracle Utilities Billing Component finds all selected accounts based on billing cycle dates, calculates the charges, and puts the results in a billing report. You then review the report, and accept or reject the results for each account. Oracle Utilities Billing Component updates the Bill History record and Data Repository, then issues transaction records to the output file for approved accounts.

  Note: This mode will not save data to the Data Repository until the bill is *approved* by the user.
- **Manual Start — Current/Final Bill** or **Bill Correction:** In these modes, you specifically select an account and bill period for billing. Oracle Utilities Billing Component calculates the charges and puts the results in a billing report for your review. If you approve the results, Oracle Utilities Billing Component issues a transaction record to the output file and updates the Bill History record and Data Repository

  Note: This mode will not save data to the Data Repository until the bill is *approved* by the user.

  Manual Start *cannot* be set as a default billing mode.

You must assign a billing mode to every account in the database. You can make your assignments globally across all accounts, to selected groups, and/or to individual accounts.

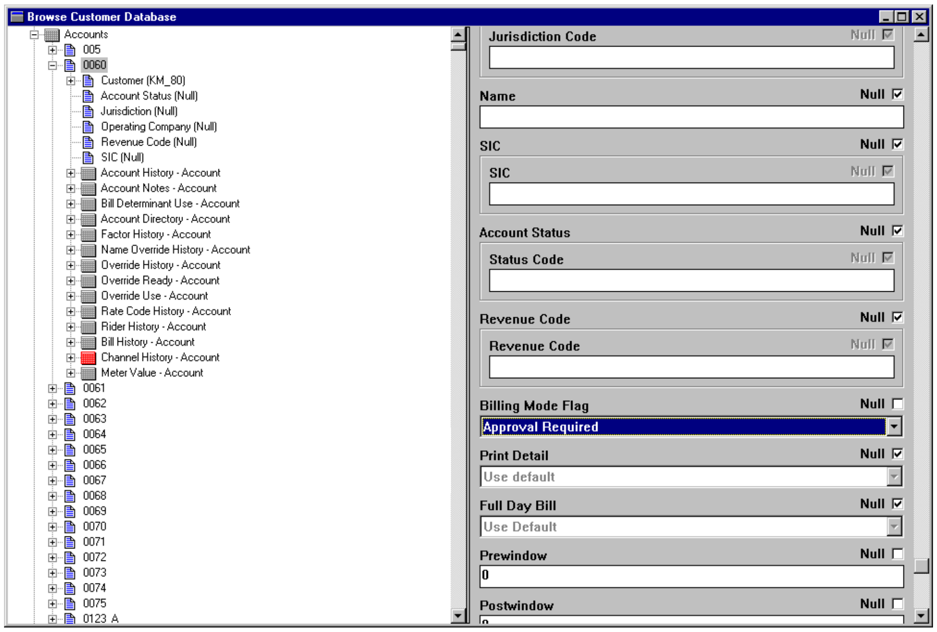
> **Note:** If a billing mode flag for any account, rate code, or rate schedule is set to anything other than **Null** or **Use Default** in the Oracle Utilities Data Repository (as described below), the value of the flag automatically overrides the global default. For example, if you set **Approval Required** as the default, and set the billing mode flag for Rate Code R1 to **Automatic Billing**, then accounts on Rate Code R1 would be billed automatically and all others (those set to **Use Default**) would require approval.

Billing Mode Flags

The Billing Mode Flags locations are:

For	In
Account	Account Table
Rate Code	Rate Codes Table
Rate Schedule	Rate Forms Table

When determining how to bill an account, Oracle Utilities Billing Component first looks at the billing mode flag in the Account record. If the flag is set to **Automatic Billing** or **Approval Required**, the system bills the account as described above. If the flag is **Use Default** or **Null**, Oracle Utilities Billing Component looks for a value in the rate code record assigned to the account. If the billing mode flag there is set to **Use Default** or **Null**, Oracle Utilities Billing Component looks for a value in the applicable rate form record. If there is a **Use Default** or **Null** value there, Oracle Utilities Billing Component uses the global default specified in the **Default Billing Options** dialog box.


Browse Customer Database

Accounts
005
0060
Customer (KM_80)
Account Status (Null)
Jurisdiction (Null)
Operating Company (Null)
Revenue Code (Null)
SIC (Null)
Account History - Account
Account Notes - Account
Bill Determinant Use - Account
Account Directory - Account
Factor History - Account
Name Override History - Account
Override History - Account
Override Ready - Account
Override Use - Account
Rate Code History - Account
Rider History - Account
Bill History - Account
Channel History - Account
Meter Value - Account
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0123 A

Jurisdiction Code — Null ✓
Name — Null ✓
SIC — Null ✓
SIC — Null ✓
Account Status — Null ✓
Status Code — Null ✓
Revenue Code — Null ✓
Revenue Code — Null ✓
Billing Mode Flag — Null ☐: Approval Required
Print Detail — Null ✓: Use default
Full Day Bill — Null ✓: Use Default
Prewindow — Null ☐: 0
Postwindow — Null ☐

Print Detail

This section describes the levels of print detail available and how to set the print detail options.

- **What Levels of Print Detail are Available?**
- **Setting the Print Detail Option**

What Levels of Print Detail are Available?

The **Print Detail** option lets you specify the level of information that Oracle Utilities Billing Component provides in both the bill reports and output file transaction records.

The specific information computed and available for reporting is determined by the statements in the applicable rate form. The Print Detail option enables you to specify how much of that available data is included in the reports and transaction records. Three levels are available:

- **None:** Effective revenue only.
- **Normal:** Effective revenue, plus any charges assigned to a revenue identifier and any values assigned to a Report Statement in the applicable rate form.
- **All:** Effective revenue, plus revenue identifiers, plus the values associated with the Rules Language Label and Report statements.

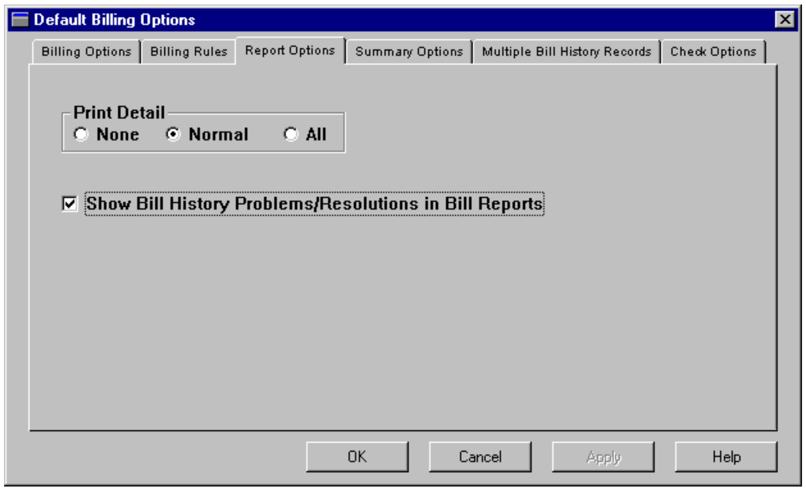
The table below details how these options apply to specific components of a report and output file transaction record. The various elements of each type of report are illustrated in **Chapter 3**: **Working with Reports** in the *Oracle Utilities Billing Component User's Guide* and **Chapter 4**: **General, Billing, and Report Options** in the *Oracle Utilities Billing Component Web User's Guide*.

Report Component	**None**	**Normal**	**All**
Customer/Account Name, ID	✓	✓	✓
Effective Revenue	✓	✓	✓
Customer/Account Detail			✓
Input Values		✓	✓
Revenue Identifiers		✓	✓
LABELed Identifier Values			✓
Simple Report Values		✓	✓
Compound Report Values		✓	✓

Setting the Print Detail Option

You can specify the print detail in a number of ways. These are described in reverse order of precedence; settings specified for a specific job take precedence over Print Detail Flags, which in turn take precedence over global default settings.

1. Set global preference in **Default Billing Options** dialog: To do so, select **CIS Billing->Bill Options**. Select the **Report Options** tab, and select the desired option under **Print Detail**.



Print Detail: None [✓ Normal] All

[✓] Show Bill History Problems/Resolutions in Bill Reports

2. Set preferences in the Oracle Utilities Data Repository. Use the Browser or Import program to supply a value in the Print Detail field for a selected rate form, rate code, or account record. You can also use the Account-Centric View for accounts.

   The Print Detail Flags locations are:

For	In
Account	Account Table
Rate Code	Rate Codes Table
Rate Schedule	Rate Forms Table

   Account settings override rate code settings, which override rate form settings.

   When determining the level of detail to include in a report or transaction record, Oracle Utilities Billing Component first looks at the Print Detail flag in the account's Account record. If the flag is set to None, Normal Detail, or All Detail, the system prints the report as described on the preceding page. If the flag is Use Default or Null, Oracle Utilities Billing Component looks for a value in the appropriate rate code record. If the billing mode flag there is set to Null or Use Default, Oracle Utilities Billing Component looks for a value in the applicable rate form record. If there is a Null or Use Default value there, Oracle Utilities Billing Component uses the global default specified in Default Billing Options.

3. Set preference for an individual job using the **Report Options** dialog. To do so, select the Report Options button in the dialog box that appears when initiating a job. See **Chapter 3**: **Working with Reports** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 4**: **General, Billing, and Report Options** in the *Oracle Utilities Billing Component Web User's Guide* for more information.

   Note: This option overrides all other settings for just the current job, and no other.

Full Day Bill

This section describes the full day bill option.

- **What is a Full Day Bill?**

What is a Full Day Bill?

Some bill charges are based on daily usage. However, the last day in the bill period is rarely a full 24-hour period. The **Full Day Bill** option lets you specify that the end of the bill period, as recorded in the Bill History record, is the last full 24-hour period (ending at midnight) before the meter read date and time. The remaining fraction of the last day is included in the next bill period. If you do not specify this option, Oracle Utilities Billing Component stores the actual read date and time as the end of the bill period.

You set the Full Day Bill option in the Oracle Utilities Data Repository, using the Browser, Import program, or (for accounts) the Account- Centric View. As with the settings described on the previous pages, you can specify the Full Day Bill option at the Account or Account History level. However, unlike the other settings, the Full Day Bill option does not have a place to set a global default — the global defaults **do not apply full day billing**.

The Full Day Bill Flags locations are:

For	In
Account	Accounts Table
Account History	Account History Table
Rate Code	Rate Codes Table
Rate Form	Rate Forms Table

Account settings override rate code settings, which override rate form settings. The options are:

- **Yes:** Apply Full Day Bill option.
- **No:** Do not apply Full Day Bill option.

Eligibility Windows

This section describes the eligibility windows used by Oracle Utilities Billing Component and how to set them.

- **What are Eligibility Windows?**
- **Setting the Pre- and Post-Windows Values**

What are Eligibility Windows?

When billing using the Automatic and Approval Required modes, Oracle Utilities Billing Component looks for accounts that are "eligible" for billing. Eligibility is determined by whether or not the account's meter is scheduled to be read on a date that is "near" today (the current date). A date is considered "near" today if today's date falls within a user-defined window that extends some number of days before and some number of days after the scheduled meter read date. The number of days before is the prewindow; the number of days after is the postwindow. An account's scheduled read dates are specified by its Billing Cycle code in its Account History record in the Oracle Utilities Data Repository.

When the account becomes eligible for billing, Oracle Utilities Billing Component begins scanning for the data necessary to calculate its bill. If the data does not become available before the eligibility window closes, Oracle Utilities Billing Component tags a SYSTEM REJECTED note to the account (see **Chapter 8**: **Review Billing Exceptions** in the *Oracle Utilities Billing Component User's Guide* or **Billing Exceptions** on page 10-5 in the *Oracle Utilities Billing Component Web User's Guide*). When that happens, you must correct the problem before issuing the bill. If the correction occurs during the eligibility window, the bill can still be issued automatically. Otherwise, you must issue the bill manually using **Current/Final Bill** (see **Chapter 7**: **Current/Final Bill** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 8**: **Current/Final Bill** in the *Oracle Utilities Billing Component Web User's Guide*).

Setting the Pre- and Post-Windows Values

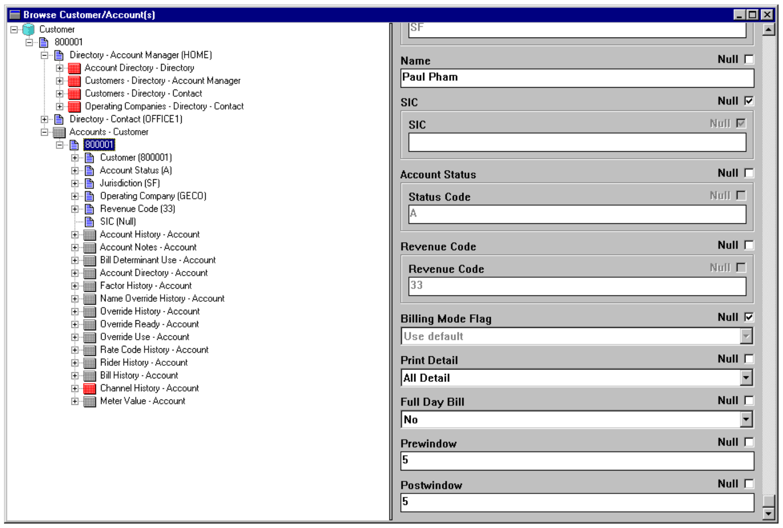
You can set the eligibility window globally for all accounts, for specific accounts, and/or specific types of recorders. In the following list, each succeeding method takes precedence over the former.

1. Set global preference in **Default Billing Options** dialog. To do so, select **CIS Billing-›Billing Options**. Then specify a number of days for the **Default Eligibility Pre-Window** and the **Default Eligibility Post-Window**.

   You can supply any value from 0 to 28 for either window. If you set the pre-window to 0, Oracle Utilities Billing Component doesn't begin scanning for data until the scheduled read date. If you set the post-window to 0, the data must be available by the end of the read date or else it is considered missing.

   Note: The Default Pre- and Post-Windows should be set the same or larger than the largest Pre- and Post-Window for any single account.

2. **Set preferences for selected accounts and/or recorder types in the Oracle Utilities Data Repository.** Use the Browser or Import program to specify the prewindow and postwindow for selected accounts in the Account Table and/or recorders in the Recorder Table. You can also use the Account-Centric View for accounts. The values in the Account Table override those in the Recorder Table.



Browse Customer/Account(s)

- Customer
  - 800001
    - Directory - Account Manager (HOME)
      - Account Directory - Directory
      - Customers - Directory - Account Manager
      - Customers - Directory - Contact
      - Operating Companies - Directory - Contact
    - Directory - Contact (OFFICE1)
    - Accounts - Customer
      - 800001
        - Customer (800001)
        - Account Status (A)
        - Jurisdiction (SF)
        - Operating Company (GECO)
        - Revenue Code (33)
        - SIC (Null)
        - Account History - Account
        - Account Notes - Account
        - Bill Determinant Use - Account
        - Account Directory - Account
        - Factor History - Account
        - Name Override History - Account
        - Override History - Account
        - Override Ready - Account
        - Override Use - Account
        - Rate Code History - Account
        - Rider History - Account
        - Bill History - Account
        - Channel History - Account
        - Meter Value - Account

	SF	
Name	Paul Pham	Null ☐
SIC		Null ✓
SIC		Null ✓
Account Status		Null ☐
Status Code	A	Null ☐
Revenue Code		Null ☐
Revenue Code	33	Null ☐
Billing Mode Flag	Use default	Null ✓
Print Detail	All Detail	Null ☐
Full Day Bill	No	Null ☐
Prewindow	5	Null ☐
Postwindow	5	Null ☐

The windows for recorder types are actually specified in the Interrogation Code Table. This means when you select the Interrogation Code, you are setting the windows. In the Interrogation Code Table, the prewindows and postwindows are both typically set to 3 days for manually read recorders, and 0 for AMR equipment.

Note: If an account has multiple recorders with different eligibility windows, Oracle Utilities Billing Component finds the smallest window specified among the recorders and applies it to all of the account's recorders.

When determining the eligibility window for an account, Oracle Utilities Billing Component first finds the default pre- and post-windows (from the **Billing Option**s tab on the **Default Billing Options** dialog), and then looks at the Account Table. If there are no values specified there, it looks in the Channel History Table to determine whether or not the Account is billed using interval data. If so, the system looks at the interrogation code associated with the channel's recorder ID to see if a prewindow and postwindow are specified. If so, the system applies it.

Setting Up Summary Bills for Customers

Some customers receive a single **Summary Bill** that combines charges for two or more accounts. To accommodate this situation, you can specify that Oracle Utilities Billing Component hold processing of the individual accounts until data for all of them is available. When all the data is available, Oracle Utilities Billing Component releases all of the transaction records to the output file for the accounts that make up the Summary Bill at the same time.

To specify that a customer receives a Summary Bill (hold account transaction records until all are ready), you must set the Summary Bill flag in the customer's record in the Customer Table to **Yes**. You can do this via the Import utility, or manually using the Browser.

The date that Oracle Utilities Billing Component calculates the account's bills and issues the transaction records is dependant upon the latest scheduled read date among all of the customer's accounts. However, each account's bill is still calculated according to its own actual read date. For example, suppose that a customer has three accounts. The scheduled read date for Account 1 is the 10th of the month, for Account 02 it's the 15th, and for Account 03 it's the 20th. Each of them has a three-day prewindow, and a one-day postwindow. In this scenario, Oracle Utilities Billing Component would begin scanning for data for all three accounts on the 17th. If all of their data came in on the scheduled read dates, all of their transaction records would be issued the 20th. However, the bill for Account 01 would end on the 10th; for Account 02 on the 15th; and for Account 03 on the 20th.

For more information about summary bills, including summary billing for master accounts, see **Summary Bill Calculation** on page 4-20 for more information.

Specifying Usage Data Required to Bill an Account

Oracle Utilities Billing Component requires up to three sets of monthly values to compute an account's bill: interval data, bill determinant values, and special event overrides. Not all three types are required for every account. This section explains how to set up an account's requirements for the first two. Special event overrides are described in **Chapter 6**: **Setting Up and Applying Overrides and Special Events**.

How to specify that an account is billed using interval data:

To link an account in the Oracle Utilities Data Repository to its cuts in the Interval Database, you must set up records in the following tables for the account and its recorders: Customer, Account, Recorder, Channel, and Channel History. For each channel that is used to bill the account, you must set the Channel History record's **Required for Billing?** field to **Yes**.

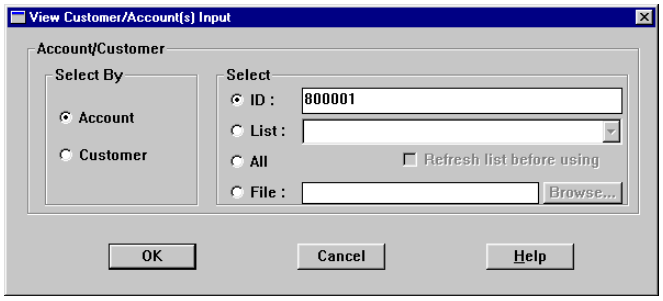
How to specify that an account is billed using bill determinants:

Bill determinant requirements are specified in the applicable rate form, but when evaluating whether an account is available for billing, Oracle Utilities Billing Component first checks the **Bill Determinant Use** Table to see what bill determinants are required.

> **Note**: The steps outlined assume that you are using the Browser. Alternatively, you can use the Import utility or the Account-Centric View.

1. Select **File->Browse->Customer/Account(s)**.

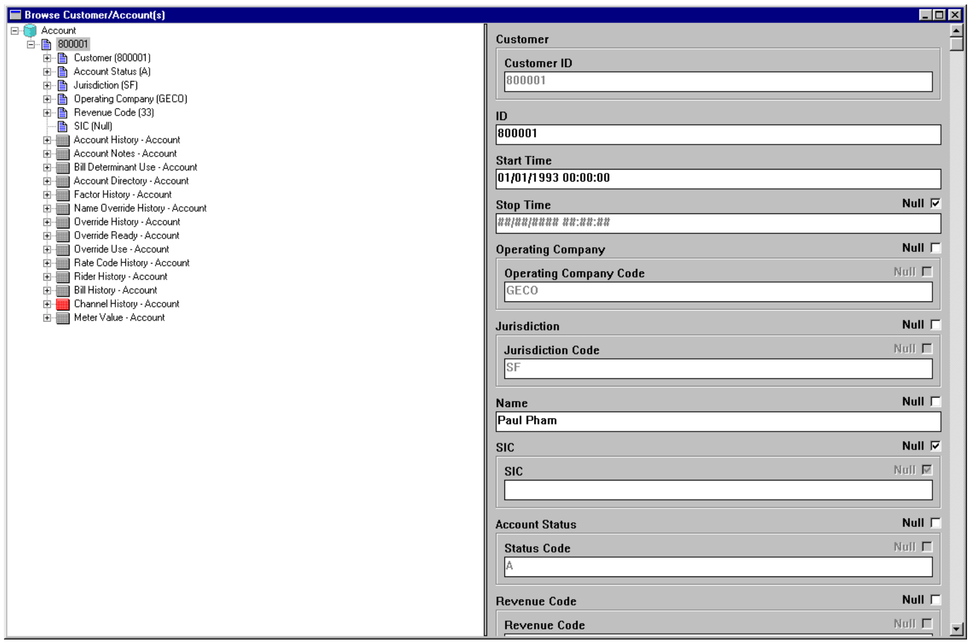
   The **View Customer/Accounts Input** dialog box appears.



Select By: [✓ Account] Customer
Select: [✓ ID :] 800001

2. Specify the ID of the desired account and click **OK**.

   The **Browse Customer/Account(s)** window appears.



Account
- 800001
  - Customer (800001)
  - Account Status (A)
  - Jurisdiction (SF)
  - Operating Company (GECO)
  - Revenue Code (33)
  - SIC (Null)
  - Account History - Account
  - Account Notes - Account
  - Bill Determinant Use - Account
  - Account Directory - Account
  - Factor History - Account
  - Name Override History - Account
  - Override History - Account
  - Override Ready - Account
  - Override Use - Account
  - Rate Code History - Account
  - Rider History - Account
  - Bill History - Account
  - Channel History - Account
  - Meter Value - Account

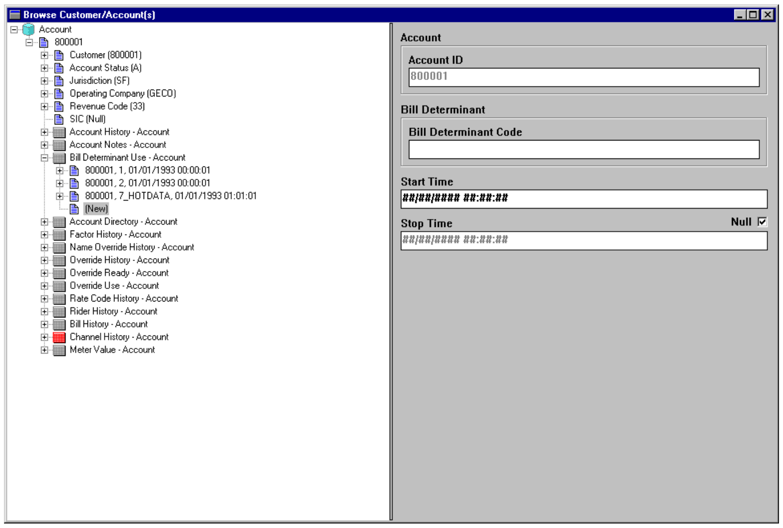
Customer
Customer ID: 800001
ID: 800001
Start Time: 01/01/1993 00:00:00
Stop Time: ##/##/#### ##:##:## Null ✓
Operating Company Null
Operating Company Code: GECO Null
Jurisdiction Null
Jurisdiction Code: SF Null
Name: Paul Pham Null
SIC Null ✓
SIC: Null ✓
Account Status Null
Status Code: A Null
Revenue Code Null
Revenue Code Null

3. Select the **Bill Determinant Use** table icon in the left pane.

   A list of bill determinants that have already been associated with this account appears in the right pane.

4. Select **Records->Insert New** or click the *right* mouse button.

   A blank data entry form appears.



Account
- 800001
  - Customer (800001)
  - Account Status (A)
  - Jurisdiction (SF)
  - Operating Company (GECO)
  - Revenue Code (33)
  - SIC (Null)
  - Account History - Account
  - Account Notes - Account
  - Bill Determinant Use - Account
    - 800001, 1, 01/01/1993 00:00:01
    - 800001, 2, 01/01/1993 00:00:01
    - 800001, 7_HOTDATA, 01/01/1993 01:01:01
    - (New)
  - Account Directory - Account
  - Factor History - Account
  - Name Override History - Account
  - Override History - Account
  - Override Ready - Account
  - Override Use - Account
  - Rate Code History - Account
  - Rider History - Account
  - Bill History - Account
  - Channel History - Account
  - Meter Value - Account

Account
Account ID: 800001
Bill Determinant
Bill Determinant Code:
Start Time: ##/##/#### ##:##:##
Stop Time: ##/##/#### ##:##:## Null ✓

5. Complete the form:

   Bill Determinant Code: The bill determinant required to bill the account.

   Start Time: The date and time when the requirement went into effect.

   Stop Time: The date and time when the requirement is no longer in effect. If the requirement is currently in effect, leave the Null box checked.

6. Select **Edit->Add** to save the information to the database.

7. Repeat for each determinant required to bill the account.

Applying Tariff Riders to Accounts

Tariff riders are simply riders that have been associated with a specific account in the Rider History Table of the Oracle Utilities Data Repository. In order to apply a tariff rider to an account, it is necessary to: 1) create and save the rider using the Oracle Utilities Rules Language, and 2) associate the rider with each applicable account via the Account Rider History Table.

This section explains how to perform the second step. If someone at your utility has not already created the rider, see the *Data Manager User's Guide* and the *Oracle Utilities Rules Language User's Guide* for instructions.

Note: Do not confuse tariff riders with the larger class of generic riders. Generic riders are simply sets of Rules Language statements that perform commonly used calculations or reporting functions. They are saved separately so that the rate analyst can quickly incorporate them into a rate schedule without having to write them from scratch each time. Tariff riders, on the other hand, are a sub-class of generic riders. *The practical difference between a tariff rider and a non-tariff rider is that you can easily report the application of tariff riders.* This report, called the **Rider Summary**, enables you to list the tariff riders that are or have been applied to selected accounts, or vice versa. See **Chapter 11**: **Other Reports** in the *Oracle Utilities Billing Component User's Guide* or **Chapter 10**: **Billing Reports** in the *Oracle Utilities Billing Component Web User's Guide* for instructions for running a Rider Summary report.

You designate a rider as a tariff rider when you attach it to an account by listing it in the **Account Rider History** Table, described below.

> **Note:** The following assumes that you are using the Browser to update the Account Rider History Table. Alternatively, you can use the Import Program or the Account-Centric View.

How to apply a tariff rider to an account:

1. Select **File->Browse Customer/Account(s)**.

   The **View Customer/Accounts Input** dialog box appears.

2. Specify the ID of the desired account and click **OK**.

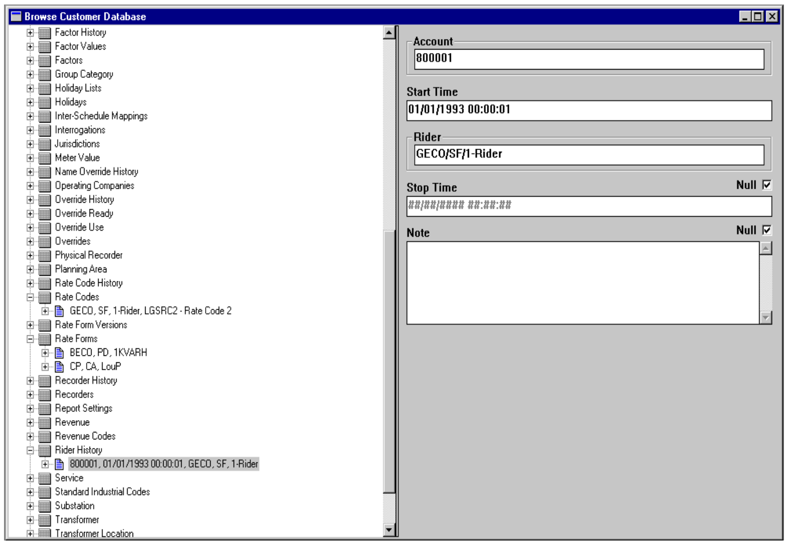
   The **Browse Customer/Account(s)** window appears.

3. Select the **Rider History** table icon in the left pane.

   A list of tariff riders that have already been associated with this account appears in the right pane.

4. Select **Records->Insert New,** or click the *right* mouse button.

   A blank data entry form appears.



Browse Customer Database

- Factor History
- Factor Values
- Factors
- Group Category
- Holiday Lists
- Holidays
- Inter-Schedule Mappings
- Interrogations
- Jurisdictions
- Meter Value
- Name Override History
- Operating Companies
- Override History
- Override Ready
- Override Use
- Overrides
- Physical Recorder
- Planning Area
- Rate Code History
- Rate Codes
  - GECO, SF, 1-Rider, LGSRC2 - Rate Code 2
- Rate Form Versions
- Rate Forms
  - BECO, PD, 1KVARH
  - CP, CA, LouP
- Recorder History
- Recorders
- Report Settings
- Revenue
- Revenue Codes
- Rider History
  - 800001, 01/01/1993 00:00:01, GECO, SF, 1-Rider
- Service
- Standard Industrial Codes
- Substation
- Transformer
- Transformer Location

Account: 800001

Start Time: 01/01/1993 00:00:01

Rider: GECO/SF/1-Rider

Stop Time: ##/##/#### ##:##:## Null ✓

Note: Null ✓

5. Complete the form:

   Start Time: The date and time when the account becomes subject to the rider.

   Rider: This is the rider to be associated to the account. To select the rider, click the word "Rider." A list of available riders will appear (if the desired rider does not appear, it has not been defined in the Rider Table). Select the desired code from the list that appears, and press ENTER or click **OK**.

   Stop Time: The date and time when the account is no longer subject to the rider. If the rider is currently in effect, leave the Null box checked.

   Note: Optional descriptive information.

6. Select **Edit->Add** to save the information to the database.

Chapter 8

Setting Up Billing to run in Batch Mode

This chapter describes how to set up billing to run in batch mode, including:

- **Automatic Billing**
- **Approval Required**
- **Current/Final**
- **Bill Correction**
- **Mass Billing**

Automatic Billing

You can run automatic billing in batch mode using the AUTOBILL.EXE command line program.

Note: Multiple instances of the AUTOBILL.EXE program can be run concurrently. See the -m and -l parameters for details.

AUTOBILL Command Syntax

The command to run AUTOBILL uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s “11/01/1999 12:00:00”). Also, when you enter the command, you must either supply the path to the program, or run it from the directory in which it is stored (typically, \LODESTAR\BIN).

autobill -f *configfilename* **-c** *connectstring* [**-q** *qualifier*] **-a** *(a|c)(a|i|l* [*] |*f) name*
-t *billdate* **-s** *donotapprove* **-1** *approveeachpage* **-m** *summaryaccount* **-n** *showunbilledaccounts*
-h *showallaccounts* **-l** *#accounttoprocess (##:##)* **-lm** *memorycheck (##:##)* **-o** *cisoutputfilename*
-d *printdetail* **-w** *reportfilename* **-rp** *reportoptions* [**-lcfg** *logging configuration filename*]

In actual use, the command must be entered on one line. Also, you must either change to the directory in which the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **autobill -?** at the command prompt.

Parameter	Description
-c	*connectstring* is database connection information for the Oracle Utilities Data Repository. This parameter is **required** and must be in one of the following formats: For Oracle databases: "Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;" where: • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID.
-f	*configfilename* is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for *configfilename*, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the *Oracle Utilities Energy Information Platform Configuration Guide*.
-q	*qualifier* is an optional database qualifier. The default is PWRLINE.

Parameter	Description
-a	*(a|c)(a|i|l [*]|f)name* Account/Customer All/Id/List/File name. The default is all accounts. An * immediately after l means refresh the list. If -aaf is specified, the file must be one of the following: • a list of customer/account IDs (*.ids) • a list file (*.lst) • an account query file (*.qra) • a customer query file (*.qrc). **Examples**: • **All Accounts**: -aaa • **Account ID**: -aai555888 • **Account List**: -aalMAC_RUNRS_ACCTLIST • **Account ID File**: -aafMAC_RUNRS_ACCTFILE.ids. This file exists in the specified user directory (defined on the User Files tab of the Default Options dialog)
-t	*billdate* is the Bill Date, in MM/DD/YYYY format. The default is the current date.
-s	*donotapprove* If you include this switch, autobill does not Approve accounts. The default is to Approve.
-1	*approveeachpage* If you include this switch, autobill approves each Account page if no error occurs. The default is to Approve. The -s switch excludes the -1 switch.
-m	*summaryaccount* If you include this parameter alone, autobill processes related summary accounts individually. The default is to process related accounts all together. You can also supply optional parameters to this switch, to allow different instances of autobill to process either non-summary or summary accounts. If you supply this parameter with a 0 (-m0), autobill processes only non-summary accounts. If you supply this parameter with a 1 (-m1), autobill processes only summary accounts (all together). See **Processing Multiple Instances of Autobill** on page 8-6 for more information.
-n	*showunbilledaccounts* If you include this parameter, the report shows all unbilled accounts. The default is to show only billed accounts. This shows only error accounts, including non-autobill accounts and warnings with summary. **Note**: You must define -n with -w in order to produce the transaction reports.
-h	*showallaccounts* If you include this parameter, the report shows all accounts. The default is to show only billing errors.
-l	*#accounttoprocess ###:* this is the maximum number of accounts to process. The default is to process all accounts.

Parameter	Description
-l	##:## designates the total number of instances of autobill and the specific number of this process. For example, -l1:3 would indicate the first of three instances. The default is 1 process. See **Processing Multiple Instances of Autobill** on page 8-6 for more information.
-lm	*memorycheck* ##:## Performs a "free memory" check. The first ## designates the amount of memory to check for (in K). The second ## is the frequency in number of accounts. For example, -lm10000:500 would check for 10 MB free every 500 accounts. The default is no check.
-o	*cisoutputfilename* is the name of the output file used by the CIS system. The default is LODESTAR.CIS. **Note**: oLGNA means use the LGNA tables for output.
-w	*reportfilename* is the filename for transaction reports [default: YYYYMMDD.HHMM]. **Note**: You must define -n with -w in order to produce the transaction reports.
-rp	*reportoptions* This switch turns on all Report Options - Problem Determination.

Parameter	Description
-d	*printdetail* is the Print Detail level (0/1/2/3) [default: 0]. If you put **-d0**, you'll see: • Account and customer information • Calculation breakdown • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Bill History • CIS information. If you put **-d1**, you'll see: • Account and customer information • Calculation breakdown (less detail) • Bill History • CIS information. If you put **-d2**, you'll see: • Account and customer information • Calculation breakdown • Bill History • CIS information. If you put **-d3**, you'll see: • Account and customer information • Account Start Time • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Calculation breakdown • Bill History • CIS information.
-lcfg	*logging configuration filename* Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named AUTOBILL.LOG in the LOG directory.

Return Codes

Return Code	Description
0	All accounts that could be billed were billed successfully.
1	Automatic Billing could not start.
2	No accounts are found ready to be billed.
99	At least one account failed to bill.

Processing Multiple Instances of Autobill

Autobill can be used to process both summary and non-summary accounts. In most cases, the number of non-summary accounts greatly exceeds the number of summary accounts. Autobill can be configured to allow the user to force an instance of Autobill to bill only summary or non-summary accounts, and to allow multiple instances of the non-summary Autobill. (If there are two instances that can bill summary accounts, one may start with one account, and the other with a different, related account, leading to great confusion.)

Two command line parameters are used to allow multiple Autobill instances. The first causes its instance to bill only summary or non-summary accounts, the second specifies the number of instances.

The first is the addition of 0 and 1 to the -m parameter (see description of the -m switch).

The second is an addition to the -l parameter that allows the user to specify the total number of instances and this ones number. (see description of the -l##:## switch).

If multiple instances of autobill are used, each one must have the -m0 parameter. One other instance should run using the -m1 parameter to bill all summary accounts. Every instance of Autobill must specify a different user directory. That directory will contain the log and output files.

Each instance run with the -l parameter will retrieve all matching accounts, and then bill the corresponding subset. If there are fewer ready accounts than instances, all accounts will be billed in instance 1.

A sample batch script for 4 total instances might look like this:

```
start autobill.exe -l1:3 -m0 -f…

start autobill.exe -l2:3 -m0 -f…

start autobill.exe -l3:3 -m0 -f…

start autobill.exe -m1 -f…
```

Approval Required

You can run approval required bills in batch mode using the APRVREQ.EXE command line program.

Note: Multiple instances of the APRVREQ.EXE program can be run concurrently. See the -m and -l parameters for details.

APRVREQ Command Syntax

The command to run APRVREQ uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s “11/01/1999 12:00:00”). Also, when you enter the command, you must either supply the path to the program, or run it from the directory in which it is stored (typically, \LODESTAR\BIN).

aprvreq -f *configfilename* **-c** *connectstring* [**-q** *qualifier*] **-a** *(a|c)(a|i|l* [*] |*f) name*
-t *billdate* **-s** *donotapprove* **-1** *approveeachpage* **-m** *summaryaccount*
-n *showunbilledaccounts* **-l** ##:## **-o** *cisoutputfilename* **-w** *reportfilename*
-rp *reportoptions* **-rd** *printdetail* [**-lcfg** *logging configuration filename*]

In actual use, the command must be entered on one line. Also, you must either change to the directory in which the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **aprvreq -?** at the command prompt.

Note: All Approval Required Flags are checked when APRVREQ runs.

Parameter	Description
-c	*connectstring* is database connection information for the Oracle Utilities Data Repository. This parameter is **required** and must be in one of the following formats: For Oracle databases: "Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;" where: • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID.
-f	*configfilename* is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for *configfilename*, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the *Oracle Utilities Energy Information Platform Configuration Guide*.
-q	*qualifier* is an optional database qualifier. The default is PWRLINE.

Parameter	Description
-a	*(a|c)(a|i|l [*]|f)name* Account/Customer All/Id/List/File name. The default is all accounts. An * immediately after l means refresh the list. If -aaf is specified, the file must be one of the following: • a list of customer/account IDs (*.ids) • a list file (*.lst) • an account query file (*.qra) • a customer query file (*.qrc). **Examples**: • **All Accounts**: -aaa • **Account ID**: -aai555888 • **Account List**: -aalMAC_RUNRS_ACCTLIST • **Account ID File**: -aafMAC_RUNRS_ACCTFILE.ids. This file exists in the specified user directory (defined on the User Files tab of the Default Options dialog)
-t	*billdate* is the Bill Date, in MM/DD/YYYY format. The default is the current date.
-s	*donotapprove* If you include this switch, aprvreq does not Approve Accounts. The default is to Approve.
-1	*approveeachpage* If you include this switch, aprvreq approves each Account page if no error occurs. The default is to Approve. The -s switch excludes the -1 switch.
-m	*summaryaccount* If you include this parameter alone, the program processes related summary accounts individually. The default is to process related accounts all together. You can also supply optional parameters to this switch, to allow different instances of approval required to process either non-summary or summary accounts. If you supply this parameter with a 0 (-m0), the program processes only non-summary accounts. If you supply this parameter with a 1 (-m1), the program processes only summary accounts (all together). See **Processing Multiple Instances of Aprvreq** on page 8-10 for more information.
-n	*showunbilledaccounts* If you include this parameter, the report shows all unbilled accounts. The default is to show only billed accounts.
-l	##:## designates the total number of instances of the program and the specific number of this process. For example, -l1:3 would indicate the first of three instances. The default is 1 process. See **Processing Multiple Instances of Aprvreq** on page 8-10 for more information.
-o	*cisoutputfilename* is the name of the output file used by the CIS system. The default is LODESTAR.CIS. **Note**: oLGNA means use the LGNA tables for output.

Parameter	Description
-w	*reportfilename* is the filename for transaction reports [default: YYYYMMDD.HHMM].
-rp	*reportoptions* This switch turns on all Report Options - Problem Determination.
-rd	*printdetail* is the Print Detail level (0/1/2/3) [default: 0]. If you put **-d0**, you'll see: • Account and customer information • Calculation breakdown • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Bill History • CIS information. If you put **-d1**, you'll see: • Account and customer information • Calculation breakdown (less detail) • Bill History • CIS information. If you put **-d2**, you'll see: • Account and customer information • Calculation breakdown • Bill History • CIS information. If you put **-d3**, you'll see: • Account and customer information • Account Start Time • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Calculation breakdown • Bill History CIS information.
-lcfg	*logging configuration filename* Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named APRVREQ.LOG in the LOG directory.

Return Codes

Return Code	Description
0	All accounts that could be billed were billed successfully.
1	Approval Required could not start.
99	At least one account failed to bill.

Processing Multiple Instances of Aprvreq

Aprvreq can be used to process both summary and non-summary accounts. In most cases, the number of non-summary accounts greatly exceeds the number of summary accounts. Aprvreq can be configured to allow the user to force an instance of Aprvreq to bill only summary or non-summary accounts, and to allow multiple instances of the non-summary Aprvreq. (If there are two instances that can bill summary accounts, one may start with one account, and the other with a different, related account, leading to great confusion.)

Two command line parameters are used to allow multiple Aprvreq instances. The first causes its instance to bill only summary or non-summary accounts; the second specifies the number of instances.

The first is the addition of 0 and 1 to the -m parameter (see description of the -m switch).

The second is an addition to the -l parameter that allows the user to specify the total number of instances and this one's number. (see description of the -l##:## switch).

If multiple instances of aprvreq are used, each one must have the -m0 parameter. One other instance should run using the -m1 parameter to bill all summary accounts. Every instance of aprvreq must specify a different user directory. That directory will contain the log and output files.

Each instance run with the -l parameter will retrieve all matching accounts, then bill the corresponding subset. If there are fewer ready accounts than instances, all accounts will be billed in instance 1.

A sample batch script for 4 total instances might look like this:

```
start aprvreq.exe -l1:3 -m0 -f…

start aprvreq.exe -l2:3 -m0 -f…

start aprvreq.exe -l3:3 -m0 -f…

start aprvreq.exe -m1 -f…
```

Current/Final

You can run current/final bills in batch mode using the CURFINAL.EXE command line program.

CURFINAL Command Syntax

The command to run CURFINAL uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s “11/01/1999 12:00:00”). Also, when you enter the command, you must either supply the path to the program, or run it from the directory in which it is stored (typically, \LODESTAR\BIN).

curfinal -f *configfilename* **-c** *connectstring* [**-q** *qualifier*] **-a** *(a|c)(a|i|l* [*] |*f) name*
-t *userspecifiedstop* **-s** *donotapprove* **-1** *approveeachpage* **-m** *nosummaryaccount*
-n *finalbill* **-o** *cisoutputfilename* **-w** *reportfilename* **-rp** *reportoptions* **-rd** *printdetail*
[**-lcfg** *logging configuration filename*]

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **curfinal -?** at the command prompt.

Parameter	Description
-c	*connectstring* is database connection information for the Oracle Utilities Data Repository. This parameter is **required** and must be in one of the following formats: For Oracle databases: "Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;" where: • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID.
-f	*configfilename* is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for *configfilename*, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the *Oracle Utilities Energy Information Platform Configuration Guide*.
-q	*qualifier* is an optional database qualifier. The default is PWRLINE.

Parameter	Description
-a	*(a|c)(a|i|l [*]|f) name* Account/Customer All/Id/List/File name. The default is all accounts. An * immediately after l means refresh the list. If -aaf is specified, the file must be one of the following: • a list of customer/account IDs (*.ids) • a list file (*.lst) • an account query file (*.qra) • a customer query file (*.qrc). **Examples**: • **All Accounts**: -aaa • **Account ID**: -aai555888 • **Account List**: -aalMAC_RUNRS_ACCTLIST • **Account ID File**: -aafMAC_RUNRS_ACCTFILE.ids. This file exists in the specified user directory (defined on the User Files tab of the Default Options dialog)
-t	*userspecifiedstop* is the User Specified Stop Date, in MM/DD/YYYY format. The default is computed.
-s	*donotapprove* If you include this switch, curfinal does not Approve Accounts. The default is to Approve.
-1	*approveeachpage* If you include this switch, curfinal approves each Account page if no error occurs. The default is to approve. The -s switch excludes the -1 switch.
-m	*nosummaryaccount* If you include this parameter alone, curfinal will not process related summary accounts. The default is to process related accounts.
-n	*finalbill* indicates that the bill is to be a final bill.
-o	*cisoutputfilename* is the name of the output file used by the CIS system. The default is LODESTAR.CIS. **Note**: oLGNA means use the LGNA tables for output.
-w	*reportfilename* is the filename for transaction reports [default: YYYYMMDD.HHMM].
-rp	*reportoptions* This switch turns on all Report Options - Problem Determination.

Parameter	Description
-rd	*printdetail* is the Print Detail level (0/1/2/3) [default: 0]. If you put **-d0**, you'll see: • Account and customer information • Calculation breakdown • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Bill History • CIS information. If you put **-d1**, you'll see: • Account and customer information • Calculation breakdown (less detail) • Bill History • CIS information. If you put **-d2**, you'll see: • Account and customer information • Calculation breakdown • Bill History • CIS information. If you put **-d3**, you'll see: • Account and customer information • Account Start Time • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Calculation breakdown • Bill History • CIS information.
-lcfg	*logging configuration filename* Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named CURFINAL.LOG in the LOG directory.

Return Codes

Return Code	Description
0	All accounts that could be billed were billed successfully.
1	Current/Final Bill calculation could not start.
99	At least one account failed to bill.

Bill Correction

You can process bill corrections in batch mode using the BILLCORR.EXE command line program.

BILLCORR Command Syntax

The command to run BILLCORR uses the following syntax. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s “11/01/1999 12:00:00”). Also, when you enter the command, you must either supply the path to the program, or run it from the directory in which it is stored (typically, \LODESTAR\BIN).

billcorr -f *configfilename* **-c** *connectstring* [**-q** *qualifier*] **-a** *(a|c)(a|i|l [*]|f) name*
-b *#billperiods* **-t** *billtype (CR|A|C|R)* **-s** *donotapprove* **-1** *approveeachpage*
-m *summaryaccount* **-h** *nobillhistory* **-v** *nobhbilldeterm* **-d** *rebillreason* **-o** *cisoutputfilename*
-w *reportfilename* **-rp** *reportoptions* [**-lcfg** *logging configuration filename*]**-rd** *printdetail*

In actual use, the command must be entered on one line. Also, you must either change the directory to the one where the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **billcorr -?** at the command prompt.

Parameter	Description
-c	*connectstring* is database connection information for the Oracle Utilities Data Repository. This parameter is **required** and must be in one of the following formats: For Oracle databases: "Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;" where: • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID.
-f	*configfilename* is the name of the configuration file that defines the working environment of the Oracle Utilities software (directs the software where to find and place the application data files, and so on). If you do not supply a value for *configfilename*, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the *Oracle Utilities Energy Information Platform Configuration Guide*.
-q	*qualifier* is an optional database qualifier. The default is PWRLINE.

Parameter	Description
-a	*(a|c)(a|i|l [*]|f)name* Account/Customer All/Id/List/File name. The default is all accounts. An * immediately after l means refresh the list. If -aaf is specified, the file must be one of the following: • a list of customer/account IDs (*.ids) • a list file (*.lst) • an account query file (*.qra) • a customer query file (*.qrc). **Examples**: • **All Accounts**: -aaa • **Account ID**: -aai555888 • **Account List**: -aalMAC_RUNRS_ACCTLIST • **Account ID File**: -aafMAC_RUNRS_ACCTFILE.ids. This file exists in the specified user directory (defined on the User Files tab of the Default Options dialog)
-b	*#billperiods* is the number of bill periods to correct. The default is 0.
-t	*billtype (CR|A|C|R)* is the Bill type: Cancel/Rebill, Adjustment, Cancel, Rebill. The default is Cancel/Rebill.
-s	*donotapprove* If you include this switch, billcorr does not Approve Accounts. The default is to Approve.
-1	*approveeachpage* If you include this switch, billcorr approves each Account page if no error occurs. The default is to Approve. The -s switch excludes the -1 switch.
-m	*nosummaryaccount* If you include this parameter, billcorr will not process related summary accounts. The default is to process related summary accounts.
-h	*nobillhistory* If you include this parameter, billcorr will not use Bill History dates and times. The default is to use them. **Note:** The -h switch only applies if dates/times can be computed from interval data.
-v	*nobhbilldeterm* If you include this parameter, billcorr will not use current Bill History determinant values. The default is to use them. **Note:** The -h switch only applies if dates/times can be computed from interval data.
-d	*rebillreason* The reason for the bill correction.
-o	*cisoutputfilename* is the name of the output file used by the CIS system. The default is LODESTAR.CIS. **Note**: oLGNA means use the LGNA tables for output.
-w	*reportfilename* is the filename for transaction reports [default: YYYYMMDD.HHMM].
-rp	*reportoptions* This switch turns on all Report Options - Problem Determination.

Parameter	Description
-lcfg	*logging configuration filename* Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named BILLCORR.LOG in the LOG directory.
-rd	*printdetail* is the Print Detail level (0/1/2/3) [default: 0]. If you put **-d0**, you'll see: • Account and customer information • Calculation breakdown • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Bill History • CIS information. If you put **-d1**, you'll see: • Account and customer information • Calculation breakdown (less detail) • Bill History • CIS information. If you put **-d2**, you'll see: • Account and customer information • Calculation breakdown • Bill History • CIS information. If you put **-d3**, you'll see: • Account and customer information • Account Start Time • Additional Information (e.g., Revenue Code, Billing Mode Flag) • Calculation breakdown • Bill History • CIS information.

Return Codes

Return Code	Description
0	All accounts that could be billed were billed successfully.
1	Bill Correction could not start.
99	At least one account failed to bill.

Mass Billing

Oracle Utilities Billing Component categorizes accounts into three groups - very complex, complex, and not-so-complex. These correspond to the three modes of billing: Manual / Current Bill, Approval Required, and Automatic Billing. All three modes support many features including customer summary bills, interval data based start and stop times, multiple rate schedules, multiple determinants, historical determinants, overrides, and so on.

The number of accounts that fall into each category can be thought of as a pyramid - few very complex accounts, some complex account, and many not-so complex accounts. However, many, many real world accounts do not require all the features provided by Automatic Billing. For example, a residential account usually has one determinant - kWh - and only one rate schedule in effect at a time. Fast billing of these accounts can be achieved by ignoring most or all the features listed above. This kind of billing is called Mass Billing, and is processed via a command line program. This section describes the mass billing process in more detail, including:

- **Mass Billing Constraints**
- **Using the MASSBILL Command Line Program**

Mass Billing Constraints

There are a number of constraints associated with mass billing related to Rate Schedules, Accounts, and Bill Processing. These are described below.

Rate Schedule Constraints

The MASSBILL.EXE command line program bills all accounts on one specific rate schedule at a time. In essence, this is equivalent to billing all accounts in a rate class. The rate schedule name (opcocode, juriscode, rate form code) is a required parameter for MASSBILL. The other parameter used by MASSBILL is a read date. MASSBILL uses the versions of the rate schedule and included riders in effect at the end of the read date when processing bills.

A rate schedule can be used for mass billing if:

1. The rate schedule does not contain the INCLUDE CONTRACT statement. Note that it can include the INCLUDE RIDER statement.
2. The rate schedule does not require interval data.
3. All of the rate schedule's determinants are in the Bill History records, and none are in the Bill History Value Table. This will be checked, it is an error if there are no determinants or some are in the Bill History Value Table. The rate schedule can use only up to 10 determinants (usually there will be just one).
4. The rate schedule does not require historical determinants or save determinants.
5. The rate schedule does not use overrides or meter values (or at least does not require checking their tables before running the rate schedule).

The rate schedule may use season schedules, save to CIS records, save to tables and load and save financial data.

The BILLINGMODEFLAG, PRINTDETAIL and FULLDAYBILL options for the account, rate code and rate schedule are all ignored. The PRINTDETAIL is none, and FULLDAYBILL is off.

If a rate schedule computation is to be used for both Mass Billing and more complex billing, the computation should be put into a rider that is included in two different rate schedules: one used for Mass Billing only, and one used for the more complex billing.

Account Constraints

All accounts related to the rate schedule (via the Rate Code History table) and with a Bill History record with a NULL Bill Time value, a matching Read Date and non-NULL determinant values will be billed. An account can be mass billed if:

1. The account is related to only one rate schedule at a time (note that this is not checked). The one rate schedule must be a mass billing rate schedule as defined above.
2. The account's billing mode is Automatic Billing (note that this is not checked. It is assumed if the account is related to a mass billing rate schedule).
3. The account does not need interval data to determine bill start and stop. This means a Bill History record must be present to begin billing. Records in the Channel History Table are not checked.
4. Each Bill History read date is the account's effective and governing date for processing that bill history record (records in the Account History Table are not checked). All determinants used by the rate schedule must be present (not-NULL) in the Bill History record.

The account may have multiple Bill History records with the same read date, as long as there is no dependency between them. In this case, they are billed from first to last.

If there is a situation, such as in the case of different rate codes, that leads to breaking these constraints, two or more rate schedules should be created. All accounts that are related to the same rate schedule must be consistent.

Bill Process Constraints

As indicated above, the usual checks performed by Automatic Billing, Approval Required, and Current/Final Bill are not performed by Mass Billing. In addition, the following apply to Mass Billing:

1. Account bill reports are not generated, except for the final grand total summary.
2. All SAVE TO CIS statements do so via the direct write option. This means that all SAVE TO CIS statements require a SECTION name and that all columns be filled in by the rate schedule; none are filled in by default.
3. The results of each bill calculation (defined as one account, one rate schedule, one bill history record) are committed individually.
4. All prewindows and postwindows are ignored. This allows the Mass Billing process to base the effective date (and thus rate schedule/rider versions) strictly on the entered read date.
5. The combination of rate schedule and read date parameters determines a unique Mass Billing run. Several MASSBILL executables may run concurrently, as long as they have different combinations of rate schedules and read dates. When combined with #4 above, this means that several may run for the same rate schedule, but with different read dates to pick up all accounts within some window.
6. If an account bills correctly, the only database change from outside the rate schedule will be to update the corresponding Bill History Bill Time value. A COMMIT will be issued after the update to commit it and any rate schedule saves.
7. Only rate schedule Warning and Abort Account Notes/Work Queue Items are saved, no programmatic notes will be generated.
8. Automatic Billing will bill Mass Billing accounts if they are not billed before it runs. To prevent this, the billing process should be arranged to run Mass Billing in all its configurations, and then Automatic Billing to handle error and non-Mass Billing accounts.
9. The Bill Type for all bills is "CURRENT".

Mass Billing Function Processing

The number of accounts to be mass billed in one run of MASSBILL.EXE may be quite large. In the other billing modes, a list of all accounts to bill is created before any bill calculation begins. In Mass Billing, there is a query to retrieve all Mass Billing accounts to bill. Each is billed as it is retrieved. Since a database COMMIT may terminate ongoing database queries (in the same thread), the query and the bill calculations will be processed in separate threads with as much concurrency as possible (for instance, the query will get the next account while the bill is being calculated for the previous account).

The query used to retrieve an account to bill also retrieves the Bill History determinants needed to bill the account. In fact, all billing information for an account is returned all at once. The table and column information available at the start of the rate schedule include:

Table	Column
Account	Account ID
Account	Start Time
Account	Operating Company
Account	Jurisdiction
Rate Code History	All columns
Bill History	All DATETIME columns
Bill History	Bill Month
Bill History	All determinants used by the rate schedule

Note that the Account History Table is completely skipped; the Read Date in the Bill History record is assumed to be correct. Only the three tables listed above are touched by the account selection query.

Using the MASSBILL Command Line Program

As noted above, mass billing is processed in batch mode using the MASSBILL.EXE command line program.

MASSBILL Command Syntax

The command to run MASSBILL uses the syntax shown below. Parameter switches are case insensitive; you can enter them in either upper or lower case (-c or -C). If a parameter includes a space, you must enclose it in quotes (for example, -s "11/01/1999 12:00:00"). Also, when you enter the command, you must either supply the path to the program, or run it from the directory in which it is stored (typically, \LODESTAR\BIN).

massbill -f *configfilename* **-c** *connectstring* [**-q** *qualifier*] **-v** *OPCO*:*JUR*:*RS*[:VER]
-t *readdate* **-t##** *readdateaftertoday* **-t-##** *readdatebeforetoday* **-o** *cisoutputfilename*
-w *reportfilename* **-s** *donotsave* [**-lcfg** *logging configuration filename*]

In actual use, the command must be entered on one line. Also, you must either change to the directory in which the program is stored (typically, \LODESTAR\bin) before entering the command, or specify the path in the command. To view a list of all parameters on-screen, type **massbill -?** at the command prompt.

Parameters	Description
-c	*connectstring* is database connection information for the Oracle Utilities Data Repository. This parameter is **required** and must be in one of the following formats: For Oracle databases: "Data Source=<data_source>;User ID=<user_id>;Password=<password>;LSProvider=ODP;" where: • <data_source> is the Oracle TNS Name for the data source, from the TNS_NAMES.ora file (typically located in the \\<machine>\oracle\network\admin directory) • <user_id> is the user ID for the database connection • <password> is the password for the supplied user ID.
-f	*configfilename* is the name of the configuration file that defines the working environment of the Oracle Utilities software (e.g., directs the software where to find and place the application data files and so on). If you do not supply a value for *configfilename*, the system uses the default (LODESTAR.CFG). For information about the contents of this configuration file, see the *Oracle Utilities Energy Information Platform Configuration Guide*.
-q	*qualifier* is an optional database qualifier. The default is PWRLINE.
-lcfg	*logging configuration filename* Name of an optional logging configuration file that specifies where error and log messages are sent. If you omit this parameter, the application creates a log file named BILLCORR.LOG in the LOG directory.

-v	*OPCO*:*JUR*:*RS* is the identifier for the rate form. It is **required**. OPCO, JUR, RS are the operating company code, jurisdiction code, and rate form code that together identify the rate form. MASSBILL automatically uses the rate form that was applicable on the read date (specified using the -t switch). See the *Data Manager User's Guide* for more information about this convention for identifying rate forms.
-t	*readdate* is the Read Date, in MM/DD/YYYY format. The default is the current date.
-t##	*readdateaftertoday*. ## is the number of days after today: 1, 2, ...
-t-##	*readdatebeforetoday*. ## is the number of days before today: 1, 2, ...
-o	*cisoutputfilename* is the name of the output file used by the CIS system. The default is LODESTAR.CIS. **Note**: oLGNA means use the LGNA tables for output.
-w	*reportfilename* is the filename for transaction reports [default: YYYYMMDD.HHMM].
-s	*donotsave* If you include this switch, massbill does not save. Only a summary report is generated. This is used for testing [default: Save Data].

Return Codes

Return Code	Description
0	All accounts that could be billed were billed successfully.
1	Mass Billing could not start.
99	At least one account failed to bill.

Examples

Below are examples of a mass bill rate schedule and corresponding CIS format file:

Rate Schedule

```
/* Customer charge */
$CUST_CHARGE = $5.00;

/* Compute energy charges */
ALL KWH CHARGE $0.01 INTO $DISCO_CHARGE;
ALL KWH CHARGE $0.02 INTO $TRANS_CHARGE;
ALL KWH CHARGE $0.05 INTO $GEN_CHARGE;

/* Compute total charge */
$EFFECTIVE_REVENUE = $CUST_CHARGE + $DISCO_CHARGE + $TRANS_CHARGE +
$GEN_CHARGE;

/* Create and write CIS record */
CISREC.ACCTID = ACCOUNT.ACCOUNTID;
CISREC.STARTTIME = BILL_START;
CISREC.STOPTIME = BILL_STOP;
CISREC.BM = BILL_PERIOD;
CISREC.KWH = KWH;
CISREC.CUSTCHRG = $CUST_CHARGE;
CISREC.DISCO = $DISCO_CHARGE;
CISREC.TRANS = $TRANS_CHARGE;
CISREC.GEN = $GEN_CHARGE;
CISREC.TOTAL = $EFFECTIVE_REVENUE;
SAVE CISREC TO CIS SECTION "MASS_BILL_1";
```

CIS Format File

```
[SEPARATOR];,;

[SECTION]; 10
==========

[SECTION]  MASS_BILL_1
ACCTID; PIC X
STARTTIME; PIC X
STOPTIME; PIC X
BM; PIC X
KWH; PIC 9.99
CUSTCHRG; PIC 9.99
DISCO;  PIC 9.99
TRANS;  PIC 9.99
GEN;    PIC 9.99
TOTAL;  PIC 9.99
```

Chapter 9

Configuring Oracle Utilities Billing Component Security

This chapter describes how to configure security to work with the Oracle Utilities Billing Component web application, including:

- **Oracle Utilities Billing Component Security**

Oracle Utilities Billing Component Security

This section describes the securable features of Oracle Utilities Billing Component Web.

Billing Features

Oracle Utilities Billing Component web features include the following:

- **Calculations**: Enables the Calculations menu option.
    - **Bill by File**: Enables the “File” option in the Bill by drop-down list on billing calculations and reports screens.
    - **Bill by List**: Enables the “List” option in the Bill by drop-down list on billing calculations and reports screens.
    - **All**: Enables the “All” option in the Bill by drop-down list on billing calculations and reports screens.
    - **Approval Required**: Enables the Approval Required Billing function and menu option.
    - **Automatic Billing**: Enables the Automatic Billing function and menu option.
    - **Bill Correction**: Enables the Bill Correction function and menu option.
    - **Current/Final Bill**: Enables the Current/Final Bill function and menu option.
    - **Trial Calculation:** Enables the Trial Calculation function and menu option.
- **Reports**: Enables the Reports menu option.
    - **Billing Checklist Report**: Enables the Billing Checklist Report function and menu option.
    - **Billing Exceptions Report**: Enables the Billing Exceptions Report function and menu option.
    - **Billing History Report**: Enables the Billing History Report function and menu option.
    - **Late Billing Report**: Enables the Late Billing Report function and menu option.
    - **Rider Summary Report**: Enables the Rider Summary Report function and menu option.
    - **Special Events Report**: Enables the Special Events Report function and menu option.
    - **Top Accounts Report**: Enables the Top Accounts Report function and menu option.
- **Run Reports**: Enables the Run Reports option on the Billing menu.
- **View Reports**: Enables the View Reports option on the Billing menu.
- **Account Component** permissions include:
    - **Billing**: Enables the Billing clip and screen on the Account Summary screen.
        - **Insert**: Enables the Add functions on the Billing screen.
        - **Update**: Enables the edit functions on the Billing screen.
        - **Remove**: Enables the Delete functions on the Billing screen.
- **Options**: Enables the Billing Options function.

Important Notes about Assigning Oracle Utilities Billing Component Permissions

By default, the Oracle Utilities Billing Component features restrict access to all Oracle Utilities Billing Component functions and screens. To allow users access to Oracle Utilities Billing Component functionality, create users and groups and enable appropriate permissions for each.

Part Three

Contract Management Configuration

Part Three describes configuration of the contract management functionality of Oracle Utilities Billing Component, and contains the following chapters:

- **Chapter 10**: **Contract Management Rules Language Rate Schedules, Riders, and Lists**
- **Chapter 11**: **Setting Up Contract Management Database Tables**
- **Chapter 12**: **Setting Up Terms, Products, and Contract Types**
- **Chapter 13**: **Adding Customized Contract Management Business Logic**
- **Chapter 14**: **Configuring Contract Management Security**

Chapter 10

Contract Management Rules Language Rate Schedules, Riders, and Lists

This chapter describes the Rules Language rate schedules, riders, and lists used by the contract management functionality of Oracle Utilities Billing Component, including:

- **Rate Schedule/Rider Format**
- **Contract Management Rate Schedules**
- **Contract Management Riders**
- **Contract Management 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 LODESTAR. Changes made to these can result in errors when using Oracle Utilities Billing Component - Contract 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 Management Rate Schedules

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 LODESTAR. Changes made to these rate schedules can result in errors when using Oracle Utilities Billing Component - Contract 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 10-24 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

Column	Value
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*

Column	Value
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 10-24 and **Calculate Price Rider** on page 10-26 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 10-24 and **Calculate Custom Terms Rider** on page 10-26 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 10-26 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

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

Column	Value
EXTCONTRACTSTATUSCODE	No Change
SUBMITTIME	No Change
APPROVETIME	No Change
EXECUTETIME	No Change
TERMINATETIME	Current DateTime
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	TERMINATEDCONTRACT
USERID	Current UserID
DETAILS	Set Contract Status to TERMINATED

Contract Management Riders

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 Billing Component - Contract Management rate schedules.

Important Note!

Do not delete the Add History rider. **Modifications made to this rider are not supported by LODESTAR.** Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 LODESTAR.** Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 10-26 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 13**: **Adding Customized Contract Management Business Logic** for more information.

Important Note!

Do not delete the Load Attributes rider. **Modifications made to this rider are not supported by LODESTAR.** Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 13: Adding Customized Contract Management Business Logic** for more information.

Important Note!

Do not delete the Load Item Attributes rider. Modifications made to this rider are not supported by LODESTAR. Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 13: Adding Customized Contract Management Business Logic** 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 13: Adding Customized Contract Management Business Logic** for more information.

Important Note!

Do not delete the Calculate Price rider. Modifications made to this rider are not supported by LODESTAR. Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 13: Adding Customized Contract Management Business Logic** for more information.

Important Note!

Do not delete the Share Item Terms rider. Modifications made to this rider are not supported by LODESTAR. Changes made to this rider can result in errors when using Oracle Utilities Billing Component - Contract 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 13**: **Adding Customized Contract Management Business Logic** 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 13**: **Adding Customized Contract Management Business Logic** 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 13**: **Adding Customized Contract Management Business Logic** for more information.

Rate Form Name

LSCM_TERMINATE_CONTRACT_RDR

Contract Management Lists

The Oracle Utilities Billing Component - Contract Management 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 Billing Component - Contract 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

Chapter 11

Setting Up Contract Management Database Tables

This chapter describes how to set up records in a number of tables in the Oracle Utilities Data Repository used by the contract management functionality of Oracle Utilities Billing Component, 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 Oracle Utilities Billing Component. 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 specific type of 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 Billing Component - Contract 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 Billing Component - Contract 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 12

Setting Up Terms, Products, and Contract Types

This chapter describes how to set up terms and products used when creating 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 Billing Component application. See **Contract Types** on page 12-16 in the *Oracle Utilities Billing Component User's Guide* for more information about setting up Contract Types.

Setting Up Terms

Terms are specific conditions and/or circumstance required to perform contract calculations. For example, a 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.
- **Description**: A description of the term status.

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 contract calculations. Records in this table include the following information:

- **Term Type Code**: A unique code that identifies the term type.
- **Description**: A description 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 contract calculations. Records in this table include the following information:

- **Start Time**: The start time of the term in Oracle Utilities Billing Component - Contract Management. This is the time after which the term is available for use in contract calculations.
- **Stop Time**: The stop time of the term in Oracle Utilities Billing Component - Contract Management. This is the time before which the term is available for use in contract 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 Billing Component - Contract 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 Billing Component - Contract 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 Billing Component - Contract 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**.
- **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 Billing Component - Contract 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 Billing Component - Contract 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 Billing Component - Contract 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 Billing Component - Contract 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 Billing Component - Contract 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 11-2 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 11-6** 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 for the contract type (only used with contract types that require an approval process). See **Contract Type Approvals** for more information.

  Note: Work Queue Types used for contract approval processes must have the Approval Only column set to “Yes”.
- **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 13-3 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 11-4 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 11-4 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 12-2 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 functionality provided with 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.

Note: Work Queue Types used for contract approval processes must have the Approval Only column set to “Yes”.

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 **Chapter 5**: **Configuring Work Queues** of the *Oracle Utilities Energy Information Platform Installation and 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 13

Adding Customized Contract Management Business Logic

This chapter describes how to add customized business logic to the contract management functionality of Oracle Utilities Billing Component, including:

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

Customizing Contract Processes

Each of the processes performed by Oracle Utilities Billing Component - Contract 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 10**: **Contract Management Rules Language Rate 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 LODESTAR) 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. LODESTAR 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. LODESTAR 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 Billing Component - Contract Management application. See **Basics Tab** on page 12-18 of the *Oracle Utilities Billing Component 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).

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 13-5 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 13-5). 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 13-5 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 13-5 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 13-5), 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 13-5 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 13-5 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 13-5). 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 13-5 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 13-5 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 13-5), 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 13-5 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 13-5 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 13-5). 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 13-5 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 13-5 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 13-5), 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 13-5 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 13-5 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 13-8) 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 13-5 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 13-13) 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 13-5 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 13-18) 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 Contract 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 reports template (supplied with your installation of Oracle Utilities Billing Component - Contract 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 Billing Component application.

Contract Management Report Templates

The two report templates used by the contract management functionality of Oracle Utilities Billing Component 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 Billing Component 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 13-33 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 13-36 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 13-37 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 13-38 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 13-38 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 13-41 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 13-45 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 13-36 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 13-48 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 13-49 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 13-49 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 persmission 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 “attribute” 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 13-41 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 14-2 for more information about available contract management features.

Chapter 14

Configuring Contract Management Security

This chapter describes how to configure LODESTAR Security to work with the Oracle Utilities Billing Component - Contract Management application, including:

- **Oracle Utilities Billing Component - Contract Management Security**
- **Contract Management Approvals**

Oracle Utilities Billing Component - Contract Management Security

This section describes the securable features of the Oracle Utilities Billing Component - Contract Management application, including:

- **Contract Management Features**

Contract Management Features

Contract Management features include:

- **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 on the Approvals 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 Work Queue Detail on the Contract screen.
  - **Actions**: Enables the Actions drop-down list on the Contract screen.
    - **Update**: Enables the Update action on the Contract screen.
    - **Save Contract Report**: Enables the Save Contract Report action on the Contract screen.
    - **Calculate Terms**: Enables the Calculate Terms action on the Contract screen.
    - **View Calculations**: Enables the View Calculations action on the Contract screen.
    - **View Contract**: Enables the View Contract action on the Contract screen.
    - **Ready for Approval**: Enables the Ready for Approval action on the Contract screen.
    - **Submit Contract**: Enables the Submit Contract action on the Contract screen.
    - **Execute Contract**: Enables the Execute Contract action on the Contract screen.
    - **Terminate Contract**: Enables the Terminate Contract action on the Contract screen.
    - **Revise Contract**: Enables the Revise Contract action on the Contract screen.
- **Run Reports**: Enables the Run Reports menu option on the Pricing and Contracts menu.
- **View Reports**: Enables the View Reports menu option on the Pricing and Contracts menu.

Important Notes about Assigning Contract Management Permissions

By default, the Oracle Utilities Billing Component permissions restrict access to all Oracle Utilities Billing Component functions and screens. To allow users access to the contract management functionality, create roles and enable appropriate permissions for each.

Contract Management Approvals

The Approvals functionality of Oracle Utilities Billing Component uses 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 Billing Component - Contract Management, you must also apply the appropriate security to the Work Queue Open Items table to allow Oracle Utilities Billing Component - Contract 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 Installation and Configuration Guide* for detailed information about securing Work Queues items.

Part Four

Appendices

Part Four contains the following appendices:

- **Appendix A**: **Creating a CIS Transaction Record Output File**
- **Appendix B**: **Mapping Rate Codes Between Rate Schedules**
- **Appendix C**: **Oracle Utilities Data Repository - Contract Management Database Schema**

Appendix A

Creating a CIS Transaction Record Output File

The SAVE x to CIS statements in a rate form write selected results of billing or settlement calculations to an output file, which is in turn read by a Customer Informatin System (CIS) and/or other company system. The Transaction File, described in this chapter, tells the programs how to format the contents of this output file so that it is readable by the CIS or other system. The Transaction File must include separate formatting instructions for each CIS transaction type that is used at your company; for example, simple bills, complex bills, cancel/rebills.

Note: The amount of information that Oracle Utilities Billing Component writes to the output file is also affected by the **Print Detail Option** setting (**Default**, **None**, **Normal**, or **All**).

Transaction Type Description File Format

This chapter explains how to set up the Transaction File. This file is required if you wish to output the results of a bill or settlement calculation, and then transfer those results to your CIS and/or other company system.

The Transaction File Format provides you the flexibility to output this information with the Oracle Utilities Rules Language and a user-defined custom configuration output format.

After the bill or settlement information is calculated in Oracle Utilities Billing Component or in Oracle Utilities Load Profiling and Settlement, the results can be formatted and written to an output file for your CIS to process.

There are two primary files needed in order to generate and to transfer your results. One is an Output File and the other is a Control File.

Output File

You must specify the name of the output file in the LODESTAR.CFG file, an entry — CISFILENAME = D:\USER\LODESTAR.CIS — to define where the Oracle Utilities application will place the results generated by the defined fields in the control file and Rate Schedule that will be transferred to your CIS system. The user must define and specify full path for this entry in the LODESTAR.CFG file.

> **Note:** The name of the output file is user-defined. If left undefined, the default is LODESTAR.CIS.

You can also output results to a specific CIS file by using the Optional File Name in the SAVE TO CIS statement (see the *Oracle Utilities Rules Language User's Guide*). SAVE TO CIS statements that do not specify a File Name will write records to the default CIS file (specified in the LODESTAR.CFG file).

Important Note

Detail record types (31, 41, 51, and 61) can only be saved to the default CIS file, and **CANNOT** be saved to specific CIS files. See **Record Types** for more information.

Control File

You must create a Transaction File, that is a **control file** that identifies any information that you may wish to transfer to your CIS or other company system. The default name of this file is "CISFORMT.TXT". By default the name of this file shuld be defined in the LODESTAR\CFG directory. The control file is an ASCII text file, that contains a description of each CIS transaction type.

To designate a control file, include the line CISFORMAT = (*path and filename*) in the configuration file. If unspecified, the system will look for a file named CISFORMT.TXT in the LODESTAR\CFG directory. In the LODESTAR.CFG file, an entry can be added to define where the software can locate the control file (for example, "CISFORMAT = D:\USER\APPB.CTR"). To specify a control file at the rate schedule level (a file that will be used for the currently executing rate schedule), using the LSRSENV.CISFORMT_FILENAME parameter (for example, "LSRSENV.CISFORMT_FILENAME = ALTCTL.TXT").

You must define each transaction type within a Rate Schedule, and the CIS transactions are generated via the **SAVE x TO CIS** Rules Language statement within the schedule (see the *Oracle Utilities Rules Language User's Guide*).

Note: The following applies only when x is a stem identifier that was referenced earlier in the rate form. If x is a simple identifier with a string value, the string is written to the CIS file exactly as is, with no formatting, leading or trailing separators, etc.

In addition, the SAVE TO CIS statement provides the capability to specify an Optional Section Name that has been defined in the CISFORMT.TXT file. (For more information on the SAVE TO CIS statement, see the *Oracle Utilities Rules Language User's Guide*.)

For each transaction type, the control file defines all the Field Labels and their formats. The formats used in the file are Standard COBOL format definitions. The control is an ASCII text file that the user can modify in the future, if needed, using any text editor. Blank/empty lines will be allowed anywhere in the file to improve the readability of the file. If a line starts with a semi-colon (";") the line is ignored - it can be used as a comment.

This control file is very flexible and allows you to define a variable number of sections for different output formats (CIS transaction type).

In the first section of the control file, you can define control keywords that affect the format of the output file. For example, you can specify the number of sections, the number of identifiers per section, and the header by these control keywords.

The control keywords are:

Keyword	Meaning
[NOLEADINGZERO]	Do not display leading zeros in numeric fields.
[NOLABEL]	Do not display the identifier names in the output file.
[NODETAIL]	Do not include detail records in the output file.
[DETAILLINECT]	##, where ## is the number of lines per CIS record. The default is 70.
[NOTRAILSEP]	Do not include a trailing separator in each record.
[SEPARATOR];,;	Use the character between the semicolons as the column separator. To specify a non-printing character, such as a Tab or Carriage Return, use the decimal ASCII code value for the character with the following format: single quote - escape ("\") - ASCII code - single quote Examples: Tab - [SEPARATOR];'\9'; Carriange Return - [SEPARATOR];'\13';
[NOSEPARATOR]	Do not use a separator at all. This requires that all fields have fixed width. It overrides [SEPARATOR] if it comes second, otherwise the [SEPARATOR] value will be used.
[INITIALIZE]	Initialize missing values for numeric fields to zero, and missing string values to a single space (the actual field will be filled according to its format). Otherwise, if the width is fixed it is filled with spaces, if it is not fixed (there is a separator) the field is empty.
[INVOICENUMBER]	Indicates that approved bills require invoice numbers.
[STRICTTYPECHECK]	Verify that the identifier has the same type as the format string. If this is not used the identifier value is converted to match the format type, then the value is formatted. If this is used it is an error for a user assigned value to be a different type from its corresponding format field. Note that PIC 9 formats correspond to Integer and Float types, and PIC X formats correspond to String and Date types. All other value types will always generate an error. It is always as error for for a system assigned value to differ from its format. The system assigned values are listed below.
[HEADER];##	## is the number of identifiers/columns in the common header part of each record (optional).
[SECTION];##	## is the number of identifiers/columns in the transaction specific part of each record.
==========	End this part of the control file.

Entries in the header and sections consist of Field Label-COBOL format pairs, separated by semicolon (;). There will be one entry per line with each entry containing two columns: Field label and COBOL format. The field label is the name of the Rules Language identifier that contains the column's value. The format of the value to be written to the output file is defined using one of the following standard COBOL picture formats:

Format	Meaning
PIC X	any number of characters
PIC X(1)	one character
PIC XXX	three characters
PIC X(3)	three characters
PIC 9	one or more numbers (integer only)
PIC 9.99	one or more numbers left of decimal, the decimal point, and two digits right of the decimal (invalid if NOSEPARATOR is set)
PIC 9(1)	one number
PIC 999	three numbers
PIC 9(3)	three numbers
PIC 9(8).99	eight digits to the left of the decimal, the decimal point, and two digits to the right of the decimal
PIC 9(8).99-	the same with a trailing sign (only displayed if negative, invalid if NOSEPARATOR is set)
PIC -9(8).99	the same with a leading sign (only displayed if negative, invalid if NOSEPARATOR is set)
PIC 9(8)V99	eight digits to the left of the decimal, two digits to the right of the decimal, but no actual decimal point (field is 10 character wide)
PIC 9(8)V99S	the same with a trailing sign (+ or - displayed)
PIC S9(8)V99	the same with a leading sign (+ or - displayed).

Anything else is copied as is (no value substitution) after leading and trailing spaces are removed.

There are three types of sign characters that you can specify within the format,- '+' (plus), '-' (minus) and 'S'. '+' '-' are the same and cause a minus sign to appear if the value is negative, otherwise no character is output. 'S' causes a minus sign to appear if the value is negative, otherwise a plus sign is output. Only one should be used per line, it can be either leading or trailing. If NOSEPARATOR is used only S is valid, since the width is fixed.

Also, there are two decimal characters that you can specify within the format, - '.' (period) and 'V'. '.-' causes a period to appear in the decimal position. 'V' outputs only the numeric characters, without a decimal point.

> **Note:** If the width specified before the decimal point (the value in the parentheses) is insufficient to hold the value, the full value will be output (no left truncation). Hence fixed width fields must allow enough width to display any value for the field.

The use of [NOSEPARATOR] implies fixed width fields. Every field must have an explicit length - X(##) or 9(##)... - and only 'S' may be used as a sign character. Missing values will be blank filled. If the value is a number and it is too big for the field - number is 12345 and field is PIC 9(4), for example - the field is filled in with number signs (#### in this example).

For system assigned values, and user assigned values if STRICTTYPECHECK is used, if the format specifies a string and the corresponding identifier is not a string, the following string will be placed in the CIS file:

Error: PIC Xs value ##.##.

If the format specifies a number and the corresponding identifier is not a number, the following string will be placed in the CIS file:

Error: PIC 9s value 'XXX'.

The system-assigned identifiers that cannot be overridden by the user are:

INTFC_ACC_ID, CUST_NAME, INTFC_CUST_ID, TRANS_EFF_DATE, and INTFC_RCD_TY.

The system-assigned identifiers that can be overridden by the user are:

BILL_HIST_PERIOD, READ_FROM_DT, READ_TO_DT, CIS_BILL_TYPE, TURN_OFF_DT, REV_MONTH, and INVOICE_NUMBER.

Note that the format of system assigned dates is always the string format "YYYY-MM-DD".

Hyphens (-) are not allowed for definition of identifiers. If there is a value for an identifier in the Rate Form, the Field Label and value for that identifier will be written to the output file. For example, “STATE_TAX_AMT;500;”. If there is no value associated with the identifier in the Rate Form, only the delimiter is output. For example, “STATE_TAX_AMT;;”. The Field Label is only displayed if [NOLABEL] is not on. The ; can be changed to another character using the [SEPARATOR] keyword.

Only the Field Labels listed in the control file will be used to generate the correct output format to CIS. Any identifier in the Rate Form that does not have a corresponding Field Label will not be written to the output file.

The special label FILLER can be used to put spaces in the record. Its COBOL format should always be X(##). ## spaces will be placed in the output record. If [NOLABEL] is not set, "FILLER" will be used as the label in the record.

The header section begins with the keyword “[HEADER]” and ends with key string “==========” (ten = signs). Each line in it has a number of Field Label-COBOL format pairs.

Each section begins with the keyword “[SECTION]” and ends with key string “==========” (ten = signs). A section may have a name. If so, the name appears after the [SECTION] delimiter, on the same line. There must be one section with the name CANCEL if you want to do Cancel or Cancel/Rebill processing in Bill Correction.

If an account requires an EDI transaction (EDI column in ACCOUNT table record set to Y), and you want a separate format for the EDI transaction, do the following: Create a section, with a name, that defines the regular transaction format. Then create another section whose name is the same, but with “_EDI” appended. Put the EDI transaction formats in this section. Oracle Utilities Billing Component will automatically use the correct section for the two transaction types. If there is no EDI section, the regular section’s format will be used for the EDI format. See the last part of the example below.

The first line in a section may be a record type or a list of record types. If there are no record types there must be a section name in order for this section to be used. Subsequent lines contain the Label-COBOL format pairs in this section.

Record Types

There are several different record types, as listed below.

Record Types:

10 - Adjustment

20 - Cancel

30 - Current

40 - Rebill (one period)

50 - Rebill (multiple periods)

60 - Final bill

Record types 31, 41, 51, and 61 are detail records for the corresponding record type. For example, a record type of 31 indicates that it contains the details for a record type of 30, a current bill.

Sample CIS Transaction File

In the following example, the first three lines declare the setup of the control file. It declares that there is one common HEADER (with 6 Field Label-COBOL Format pairs) and two different format sections. There are three Field Label-COBOL Format pairs for format section one, two Field Label-COBOL Format pairs for section two, and so on. Format section one, in this example, shows how multiple record types can be associated with a format section. Format section two is only used for Bill Correct CANCEL transactions. Format sections three and four can handle special account requirements, with section four generating an EDI transaction record.

```
[HEADER];6
[SECTION];3
[SECTION];2
[SECTION];10
[SECTION];10
==========

[HEADER]
INTFC_CUST_ID;PIC X(21)
PWR_BILL_TY_CD; PIC X
TRANS_EFF_DATE; PIC X(10)
INTFC_RCD_TY; PIC XX
INTFC_ACC_ID; PIC X(21)
CUST_NAME; PIC X(21
==========

[SECTION]
RecordType;10;30;40;50
BILL_HIST_PERIOD; PIC 99
READ_FROM_DT; PIC X(10)
READ_TO_DT; PIC X(10)
==========

[SECTION] CANCEL
BILL_HIST_PERIOD; PIC 99
READ_FROM_DATE; PIC X(10)
==========

[SECTION] SPECIAL_ACCT
BILL_HIST_PERIOD; ACCT1 — ACCT1 appears as is
READ_FROM_DATE; PIC X(10)
...
==========

[SECTION] SPECIAL_ACCT_EDI
BILL_HIST_PERIOD; ACCT1EDI — ACCT1EDI appears as is
READ_FROM_DATE; PIC X(10)
...
==========
```

This control file will be read once and the information cached throughout the session. New changes to the control file will take effect upon restarting Oracle Utilities Billing Component.

Output for Record types 31, 41, 51, and 61 have special considerations:

- Data Values for each Field Label are taken directly from the Billing Engine's Report Window for each account (except account header). Hence NOTE 6 (above) does not apply for this case.
- Each line in the Report Window (except the report header) is a Data Value for the Field Label. There will be padding of blank lines, if there are less than 70 lines in the Report Window.
- Each Data Value will be blank padded to a maximum of 68 characters. Data Values with more than 68 columns will be truncated.
- Semicolon (;) is **not** allowed in Report Window, since semicolon is used as a delimiter.
- If the Billing Engine generates more than 70 lines of Data Values for an account, these Data Values will be split into two separate output records The record header for these two records will be the same since they are for the same record. Blank lines appearing in Billing Engine's Report Window are considered valid Data Values.

Print Detail

There are four levels of detail bill printing available for the user to select: **Default**, **None**, **Print Normal Detail**, and **Print All Detail**.

Level	Description
Default:	See documentation for Automatic Billing for a detailed explanation.
None:	No CIS transactional record will be generated upon clicking Approve menu item.
Print Normal:	*Default.* CIS record will be generated without detailed billing information. Detail information of interval data results (such as MAXDATE, TOTAL) will not be displayed.
Print All:	CIS record generated along with the information from the Bill Calculation section of the bill report. This option will display the details of interval data.

The default is Print Normal Detail. In this scenario, detail information of interval data results such as MAXDATE, TOTAL will not be displayed. To display the details of interval data, Print All Detail should be checked. If the Print None option is selected, there will be no print transaction records written to the output file.

Appendix B

Mapping Rate Codes Between Rate Schedules

The Interschedule Mapping Table defines “equivalencies” between rate codes in different rate schedules. This features makes it possible to substitute the “best” rate code for a customer when comparing different rate schedules, such as when applying Oracle Utilities Rate Management’s Customer Impact Analysis or Oracle Utilities Billing Component’s Trial Bill Calculation.

Purpose of the Interschedule Mapping Table

The Interschedule Mapping Table enables you to “pair” a rate code in one rate schedule with the most similar rate code in a second rate schedule.

This feature is useful when performing “what if” bill calculations or revenue analysis that apply rate schedules with rate code-dependent charges to customers currently on a different rate schedule/rate code.

For example, imagine that you want to find the result of applying your company's General Service rate to a customer currently on an Industrial rate. Further, say that the general service rate includes two different customer charges—one for customers on rate code GS-1 and another for customers on GS-2 (these charges are specified in the rate form using the SELECT RATE CODE statements). If the customer's current rate code is anything other than GS-1 or GS-2, the trial calculations will skip the customer charges entirely. However, if you map the industrial rate code to, say, the GS1 code in the Interschedule Mapping Table, the analysis program will apply the charges associated with GS1 when calculating the customer’s trial bill or revenue.

The Interschedule Mapping Table is stored in the Oracle Utilities Data Repository, where it is available for use with the Oracle Utilities Billing Component, Oracle Utilities Rate Management, and other analysis programs. You must use the Data Manager Browser to add, update, or delete entries in the table.

Some Important Rules for Mapping Codes

Important points to keep in mind when defining rate code mappings include:

- You can map a rate code to just one rate code in the second rate schedule (in other words, one rate code in the first schedule cannot be paired with multiple rate codes in the second schedule), however...
- You can map more than one rate code in the first schedule to the same rate code in the second schedule.
- Oracle Utilities Rate Management's Customer Impact Analysis automatically uses the mappings defined in the Interschedule Mapping Table.

Mapping Rate Codes

How to map a Rate Code:

1. Select **File->Browse->Customer Database**.

   The Browser appears.

2. Using the *right* mouse button, click the **Inter-Schedule Mappings** Table and the select **Insert New**.

   A blank data entry form appears in the right pane.

3. Complete the form:

   Rate Code: Click anywhere in this field to make the **Select Rate Code** list appear.

   This is a list of all rate schedules and rate codes currently stored in the Oracle Utilities Data Repository. If the list is very long, you can use the search tools at the top of the window to locate the desired rate code.

   Highlight the desired rate schedule/rate code and click **OK**. Your selection appears in the Browser's data entry form.

   Mapped Rate Code: This is the Rate Code you want to map to the first rate schedule/rate code selection. Use the same process as described above.

4. To save the mapping, select **Edit->Add,** or click the *right* mouse button and select **Add**.

   Note: Your new mapping appears in the left pane of the Browser.

5. Repeat steps 2 through 4 for as many mappings as you wish to create. When you have finished, click **File->Close**.

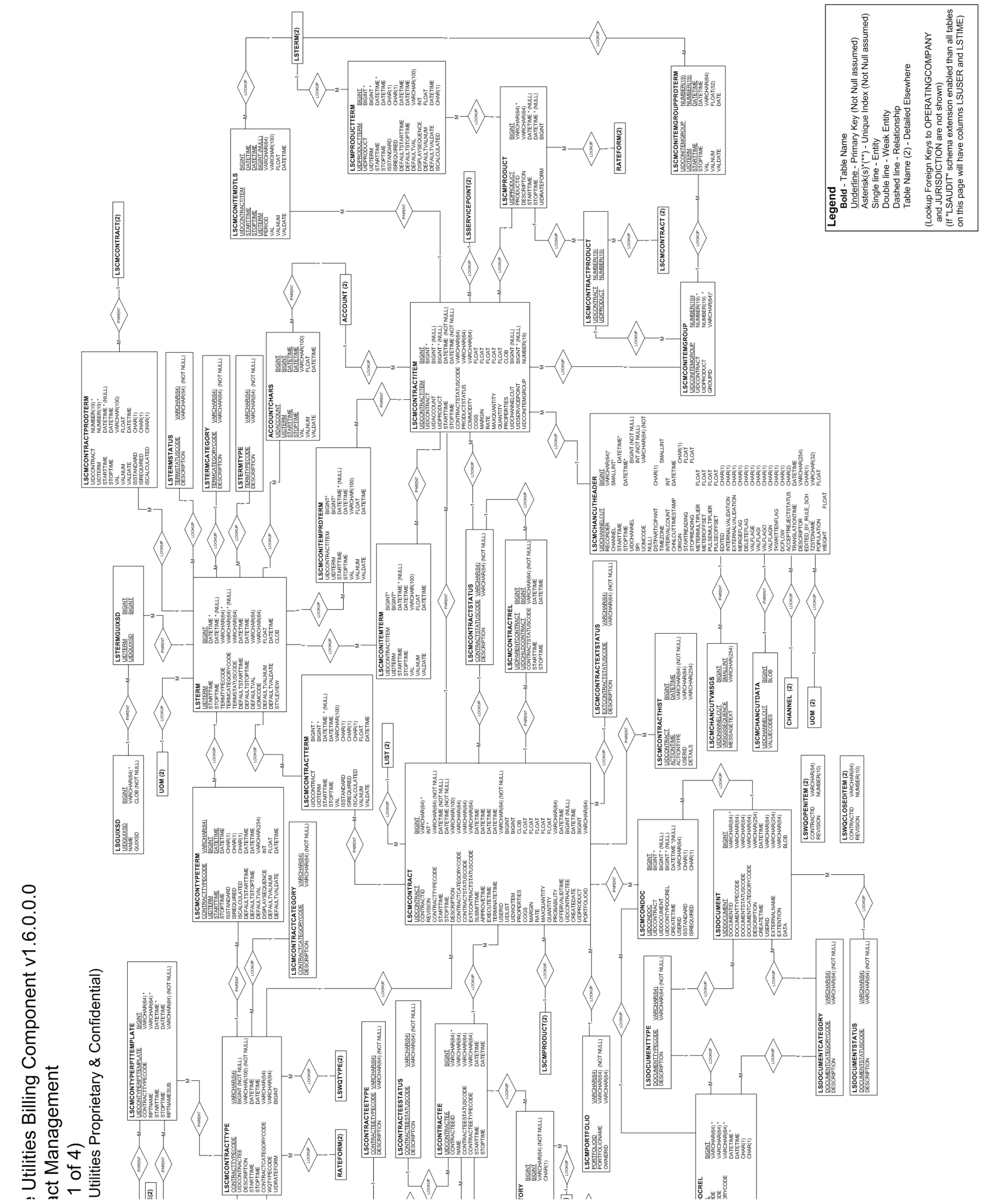
Appendix C

Oracle Utilities Data Repository - Contract Management Database Schema

This appendix includes a diagram of the Oracle Utilities Data Repository Contract Management database schema (v1.6.1.0.0) that provides details regarding the table and columns in the Oracle Utilities Billing Component - Contract 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 Billing Component - Contract Management Database Schema**

Oracle Utilities Billing Component - Contract Management Database Schema



Index